昭和59年9月1日発行(毎月1日発行)第2巻7号
昭和59年6月8日第3種郵便物認可　昭和59年2月28日国鉄首都特別扱承認雑誌第7403号

PERSONAL COMPUTER MAGAZINE
月刊オー!エフエム

Oh!FM

FM-7 NEW7 77 8 11 X

特集　どれにしようかOSいっぱい

グラフィックプログラムジェネレータ

プリンタポートで遊ぼう
簡易ジョイスティックの製作

はじめてさわるパソコン

9

1984

信頼と創造の**富士通**

マイコンスカイラブ：FMシリーズのハードからソフトまで一挙に展示実演、あなたのパソコンのコンサルタントとしてご活用ください。 ●東京・虎ノ門(03)591-1091／591-2561 ●東京・秋葉原(03)251-1448 ●札幌〈時計
ビル〉(011)222-5476〈丸井今井〉(011)241-4185 ●仙台(0222)66-8711 ●名古屋(052)221-6016 ●大阪(06)344-7628／341-0486 ●広島(082)247-3949 ●福岡〈開設準備室〉(092)471-72

天才は限度を超える。

富士通

パソコンにFM時代を拓いた富士通から、天才肌の新鋭登場。あの名作FM-7を思いきりグレードアップ。その名もFM-77です。目を見はる新機軸、息をのむ多機能、しかも豊富なソフトはそっくりそのまま継承。もう想像力に歯どめのきかないヒラメキの連続です。シンプルで知的なデザインも、一歩限度を超えました。もっとスリリングに、さらにエキサイティングに、いよいよエンターテイニングに。FM-77、いま発進です。

3.5インチマイクロフロッピィディスクを採用。
FM-77は、3.5インチのマイクロフロッピィディスクドライブを2基搭載。ディスクは、これまでの5インチミニフロッピィディスクとソフトコンパチで、しかも320KB/ドライブと、小型・高密度化を実現しました。

漢字ROM(JIS第1水準漢字2,965種)を標準実装。
レポート、手紙など、さまざまな用途に活用できます。

2個のCPUがフル稼動。
CPUの役割を2分するという、汎用コンピュータの設計思想を受け継いだアーキテクチャを採用しました。

メインメモリは64KB。増設は最大256KBまで。
64KBのメインメモリを標準実装のほか、192KB RAMカード*を装着して、最大256KBまでの拡張が可能です。
*オプション(400ラインセットに含まれています)

F-BASIC V3.0が多彩な機能をサポート。
FM-77のプログラム言語は、FM-7と完全互換のF-BASIC V3.0。豊富なソフトがそのまま使えるうえ、パレット機能、サウンド機能など数かずの機能をサポートします。

ユーザフレンドリな言語
FM Logoを標準添付。

スーパーインポーズ*で
オリジナルの映像づくり。
テレビやVTRなどの画像と、パソコンの文字やグラフィックパターンがドッキング。VTRへの録画も可能です。
*オプションのスーパーインポーズユニットが必要です。

高性能プラス多才のグラフィック機能。
FM-77は、640×200ドットの高分解能表示が魅力です。カラーモード時は、1ドットごとに8色までの色指定ができるほか、テキストとの混在表示、ドットごとの色交換もOK。400ラインカード*を本体に装着すれば640×400ドットという高精度の表示も可能になります。
*オプション(400ラインセットに含まれています)

サブシステムの充実で、画像処理がさらに高速化。
FM-77は、V RAMのアクセス方式にサイクルスチールを導入。描画速度が最高でなんと2倍(FM-7比)にアップしました。また、高速の漢字表示が、日本語ワープロや漢字端末としての利用価値を高めます。

音にも熱中。8オクターブ・三重和音のサウンド機能。

操作性抜群のキーボード。

レイアウト自在のセパレートタイプ。

大容量1MB/ドライブのフロッピィディスクをサポート。
◆F-BASIC V3.5、OS-9およびCP/M-80にてサポート。

拡張性に優れたカード群。
●400ラインセット
①400ラインカード ②増設用192KB RAMカード
③F-BASIC V3.5 ④日本語ワープロ
●各種オプションカード
RS-232C、MIDI、マウスなど。

広がるアプリケーションに対応するソフトウェア体系。
●F-BASIC V3.5(400ラインセットに含まれています)
●OS-9Level 1/Level 2*(*400ラインセットが必要です)
●CP/M-80®(Z80カードが必要です)
(※CP/M-80®は、デジタルリサーチ社の登録商標です。)

ディスクとソフトで差をつける天才パソコン。
FM-77 (セブン セブン)
¥228,000 (本体価格・FM Logo付)
ディスクドライブ1基タイプ ¥198,000もあります。

すぐれたコストパフォーマンスを実現した興奮パソコン。
FM-NEW7 (ニューセブン) **¥99,800** (本体価格)
ソフトは定評あるFM-7と完全互換、しかも数かずの卓越した機能をすべて継承しました。

富士通株式会社：●パソコン販売推進部(03)502-0161 ●札幌営業所(011)271-4311 ●東北営業所(0222)64-2131 ●金沢営業所(0762)63-7621 ●長野営業所(0262)26-8222 ●静岡営業所(0542)54-9131 ●名古屋営業所(052)201-8611 ●大阪営業所(06)344-1101 ●広島営業所(082)221-2288 ●九州営業所(092)411-6311
■3.5インチマイクロフロッピィディスクは、富士通純正品をご使用ください お問い合わせは― 富士通サフライ株式会社☎(03)434-0141

信頼と創造の富士通

マイコンスカイラブ：FMシリーズのハードからソフトまで一挙に展示実演、あなたのパソコンのコンサルタントとしてご活用ください。 ●東京・虎ノ門(03)591-1091／591-2561 ●東京・秋葉原(03)251-1448 ●札幌<時計台ビル>(011)222-5476<丸井今井>(011)241-4185 ●仙台(0222)66-8711 ●名古屋(052)221-6016 ●大阪(06)344-7628／341-0486 ●広島(082)247-3949 ●福岡<開設準備室>(092)471-7203

「通」が選んでAD2。

富士通

ハイ・マニアのハイ・パフォーマンスマシン

限りなく広がるパソコンの世界。ひとつの目標をこなして、より高度な目標に挑戦してゆく、そんなハイ・マニアのスピリットに熱くこたえるFM-11AD2。究極の8ビットCPU〝68B09E〟を、さらにパワーアップするオペレーティングシステムOS-9を採用。各モジュールの有機的なシステム化を実現しました。驚異のマルチ機能がつぎつぎと生み出す未知の体験へ、今、「通」の挑戦をお待ちします。

驚異のマルチ機能、OS-9

日本語処理機能を強化したUNIX* ライクなオペレーティングシステムOS-9Level 2を標準添付。いままで、入力、リスト、出力など一連の作業は、ひとつのモジュールの処理が終了しなければ、次の処理に進むことができませんでした。OS-9は、そういった複数のモジュールを結合、個々の処理を同時に実行させながら、各々の情報を有機的に組み合わせていきます。データの検索・入力・出力・通信などの複数の処理を同時に実行できるマルチ・タスク機能。1台のCRT画面を複数に分割し、それぞれを独立した画面として使用できるマルチ・ウィンドウ機能。1台のホストに複数のパソコンを端末装置として接続し、同時に使用することができるマルチユーザ機能など、いままでのパソコンの概念ではとらえることのできない強力な集合システムとしての稼動を実現しました。しかもOS-9上のBASICはコンパイラなので実行はあくまでもスピーディです。

1MBのミニフロッピィディスクドライブを1基標準実装

長いキャリアを持つマニアにとって、情報の蓄積は重要課題。日増しにふえる情報量に対応し、大容量ファイルの装備を実現しました。

日本語機能を強化したF-BASIC V5.0

日本語処理機能をさらに強化したF-BASIC V5.0を標準添付。このF-BASIC V5.0は、カナ/ローマ字・漢字変換方式によりBASICステートメント中の文字定数、注釈、ファイル名に漢字の使用が可能です。

JIS第1水準の漢字ROMを標準実装

JIS非漢字、JIS第1水準漢字を登録した漢字ROMを標準実装。さらにJIS第2水準漢字にも対応できます。

日本語ワードプロセッサを添付

多彩な日本語ワープロ機能を満載して、オフィスでの使用とは、また一味違う、あなただけの文章づくりが楽しめます。

多彩な機能を発揮するCRT表示

カラー表示は16色。640×400ドットのカラーグラフィックで2画面、また640×200ドットのモノグラフィックで12画面のマルチページ機能、8色までの色交換が簡単にできるパレット機能、上下左右のスクロール機能など、CRT表示機能は充実、マニアのニーズにしっかりとこたえます。

操作性重視のキーボードを採用

軽快なキー入力が可能な操作性重視のロープロファイル設計。本体との接続ケーブル(カールコード)は2.5メートルまで延長可能です。

用途を拡げる豊富なオプション群

ポインティングツールとしてのマウス、大容量20MBのハードディスクや音声合成カードなど、用途に合わせて自由なシステムアップが可能です。

*UNIXはベル研究所が開発したオペレーティングシステム(OS)の名称です。

新発売

OS-9装備の
実力派多機能パソコン。

FM-11 AD2

¥298,000 (本体価格・ワープロソフト付)

ビジネスからサイエンスまで
用途を拡げる高性能パソコン。

FM-11 BS

¥398,000 (本体価格・ワープロソフト付)

1MBの大容量5インチFDを2基搭載。本格的ビジネスに対応する16ビットマシンの決定版です。

富士通株式会社：●パソコン販売推進部(03)502-0161 ●札幌営業所(011)271-4311 ●東北営業所(0222)64-2131 ●金沢営業所(0762)63-7621 ●長野営業所(0262)26-8222 ●静岡営業所(0542)54-9131 ●名古屋営業所(052)201-8611 ●大阪営業所(06)344-1101 ●広島営業所(082)221-2288 ●九州営業所(092)411-6311

信頼と創造の富士通

スカイラブで 知性も磨く。遊び心も磨く。

富士通の
マイコンスカイラブは
FMシリーズが
ズラリ揃った
おもしろショウルーム。
見て、聞いて、触れれば、
ためになる。楽しくなる。
さあ、まじめに遊んで、
めざそう、
FMプロフェッショナル。

攻略するのに何分かかるか。ゲームに、勉強に、ソフトがいっぱいだ。

ゲームや学習ソフト、それにワープロ用や簡易言語などFMシリーズ用ソフトが数千種。どれからチャレンジするかは君次第。ビジネス用ソフトもあるから、家族で使えるよ。

何事も基本が大切。マニュアル読んで使い方のいろはをマスターしよう!

FMシリーズを自在に使いこなすには、基本の使い方を知ることが重要だ。スカイラブにはゲームや計算など、使い方に合わせたマニュアルが豊富に揃っているので、ぜひマスターしてほしい。

僕の愛読書がまた増えた。オリジナルプログラムをつくる参考にしたい。

学校の参考書はあまり読まないけど、パソコンの参考書なら、おもしろく読めるね。スカイラブには本屋に負けないくらいの本がどっさり。プログラミングのテクニックを盗んじゃおう。

富士通株式会社
富士通マイコンシステムズ株式会社

富士通

SKY-LAB
富士通マイコンスカイラブ

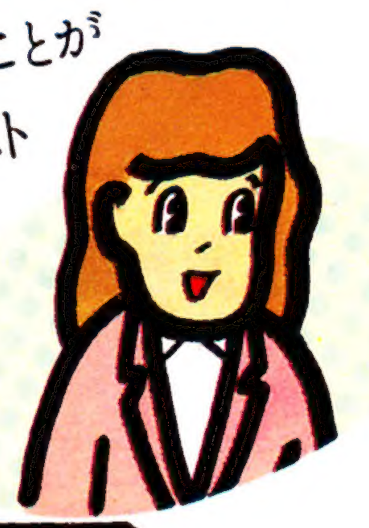
FMシリーズが勢揃い。
スカイラブがみんなを待っている。

フロッピィディスクやプリンタリボンも。FM純正品が全部揃ってる。

FMシリーズの性能を存分に引き出すには、フロッピィやプリンタリボンだって、ハードにピッタリ合ったものがいいよね。FMシリーズの純正品ならスカイラブにいっぱい。一度、見においでよ。

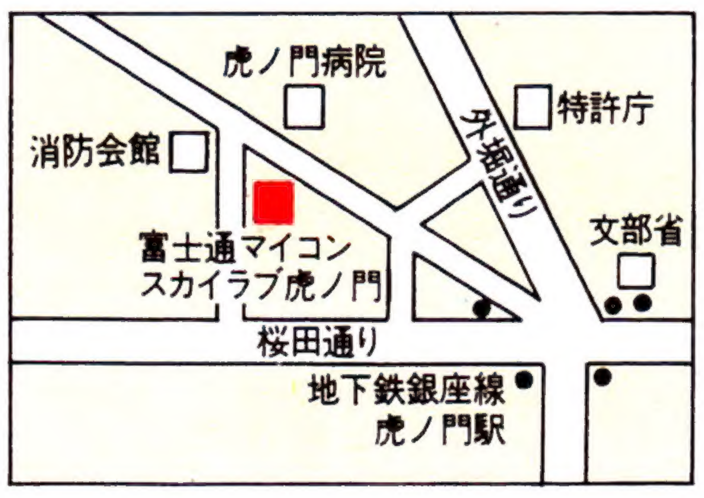
インストラクターに気軽に聞いちゃおう！FMのおもしろい使い方をいろいろ教えてくれる。

FMシリーズを使っていてわからないことがあったら、何でも相談してみよう。インストラクターは僕らの味方だからね。誰に聞いても、何を聞いても親切に教えてくれるから、心強いよ。



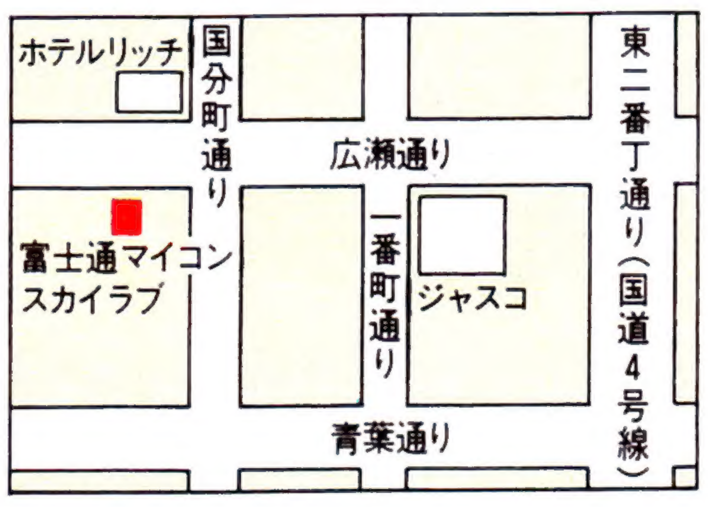
富士通マイコンスカイラブ虎ノ門

●〒105 東京都港区虎ノ門2-3-13(第18森ビル8F)
☎(03)591-1091/2561
●オープン時間：月～金 9:30～17:00(祝日をのぞく)
交通：地下鉄銀座線 虎ノ門駅下車徒歩5分



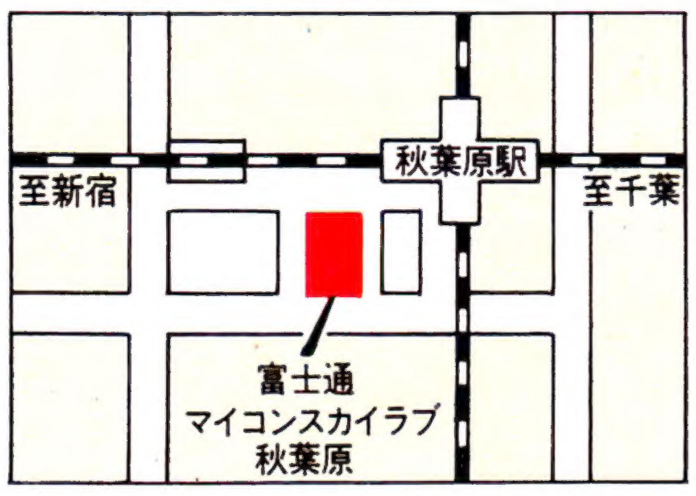
富士通マイコンスカイラブ仙台

●〒980 仙台市国分町1-7-18
(明治生命仙台国分町ビル1F) ☎(0222)66-8711
●オープン時間：10:00～19:00(木曜定休)
交通：広瀬通り ホテルリッチ向い



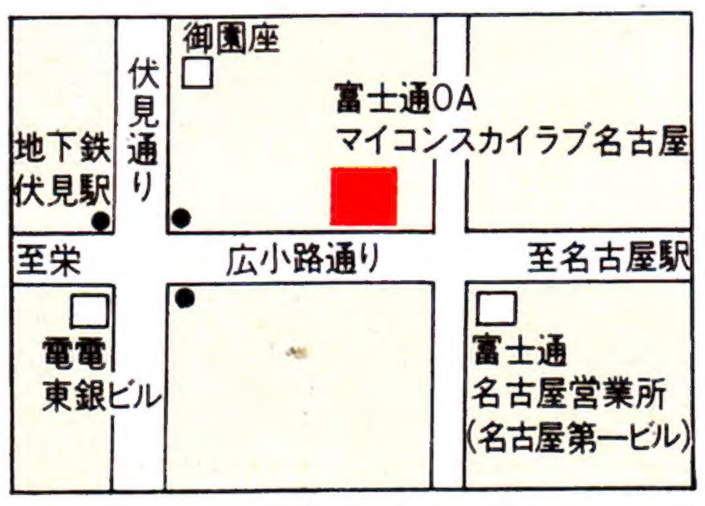
富士通マイコンスカイラブ秋葉原

●〒101 東京都千代田区外神田1-15-16(ラジオ会館6F)
☎(03)251-1448/251-1449
●オープン時間：10:00～19:00(年中無休)
交通：国鉄 秋葉原駅前



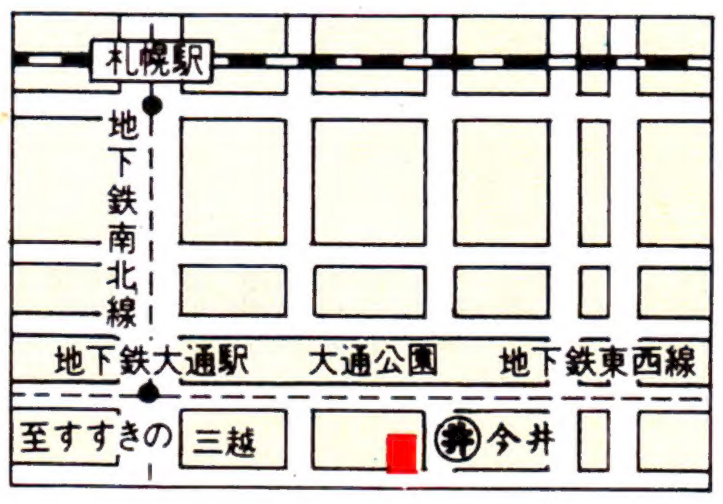
富士通マイコンスカイラブ名古屋

●〒460 名古屋市中区栄1-5-22(東ビル1F)
☎(052)221-6016
●オープン時間：月～土 10:00～18:00(祝日をのぞく)
交通：地下鉄伏見駅下車7出口より徒歩1分



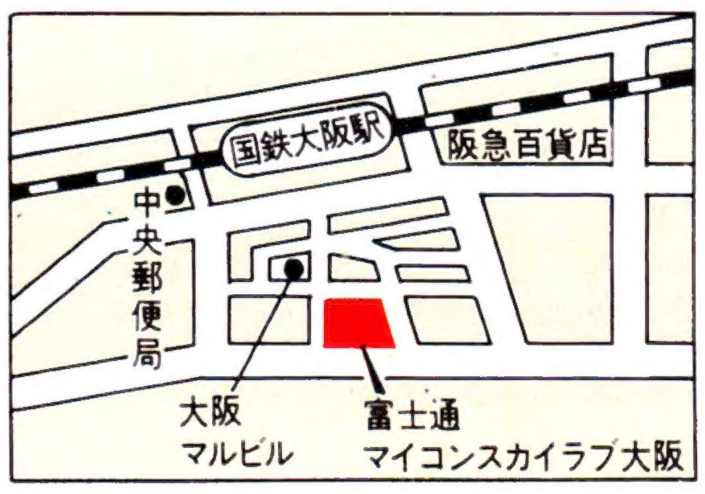
富士通マイコンスカイラブ札幌(丸井今井)

●〒060 札幌市中央区南1条西2丁目
(丸井今井1条本館4F) ☎(011)241-4185
●オープン時間：平日10:00～18:00 土・日10:00～18:30
(水曜定休)
交通：地下鉄大通駅下車



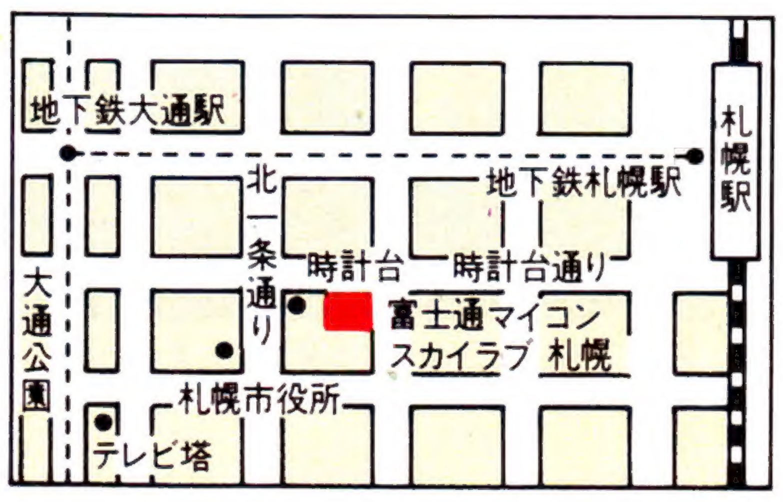
富士通マイコンスカイラブ大阪

●〒530 大阪市北区梅田1-2-2(大阪駅前第2ビル1F)
☎(06)344-7628
●オープン時間：10:00～19:00(年中無休)
交通：国鉄 大阪駅 阪神、阪急 梅田駅下車



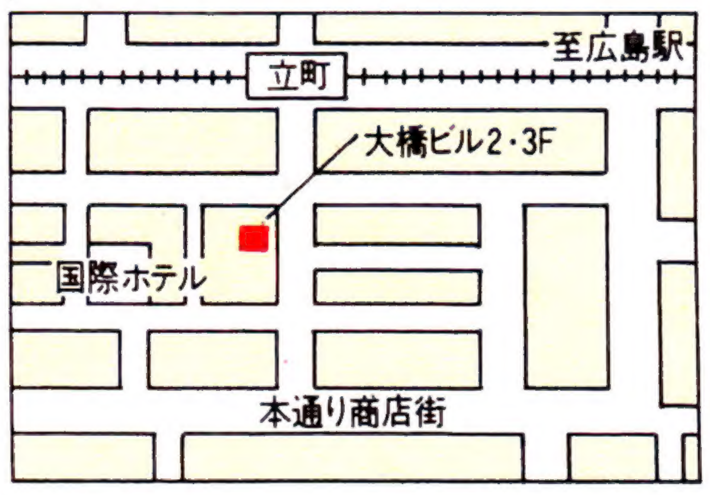
富士通マイコンスカイラブ札幌(時計台ビル)

●〒060 札幌市中央区北一条西2-1(札幌時計台ビル1F)
☎(011)222-5476
●オープン時間：月～金 10:00～18:00
土 10:00～15:00(祝日をのぞく)
交通：地下鉄 大通駅下車徒歩5分



富士通マイコンスカイラブ広島

●〒730 広島市中区立町4-2(大橋ビル2・3F)
☎(082)247-3949
●オープン時間：10:00～19:00(月曜定休)
交通：広島電鉄立町駅下車徒歩3分

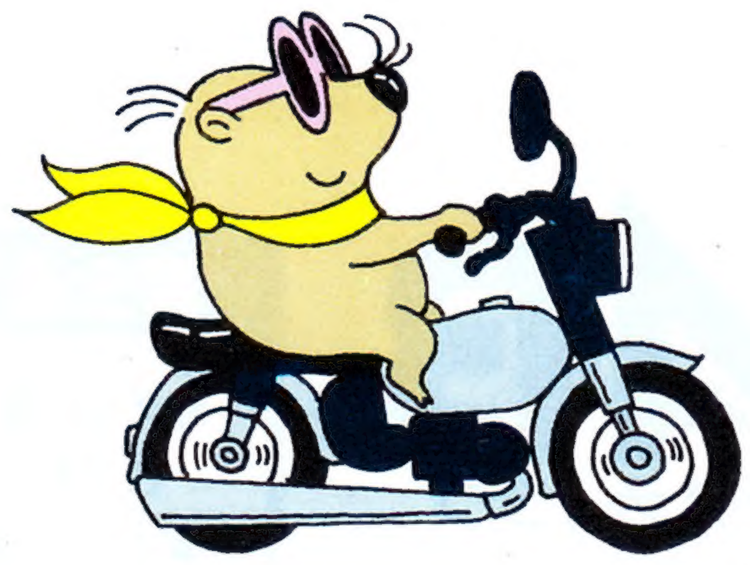


PERSONAL COMPUTER MAGAZINE

Oh!FM

1984・9

CONTENTS



特集 どれにしようか OSいっぱい

あなたOS知ってますか/田奈真実 …… 33

こんなにあるFMで使えるOS/兼子 毅 …… 38

座談会●私のOS論 …… 46

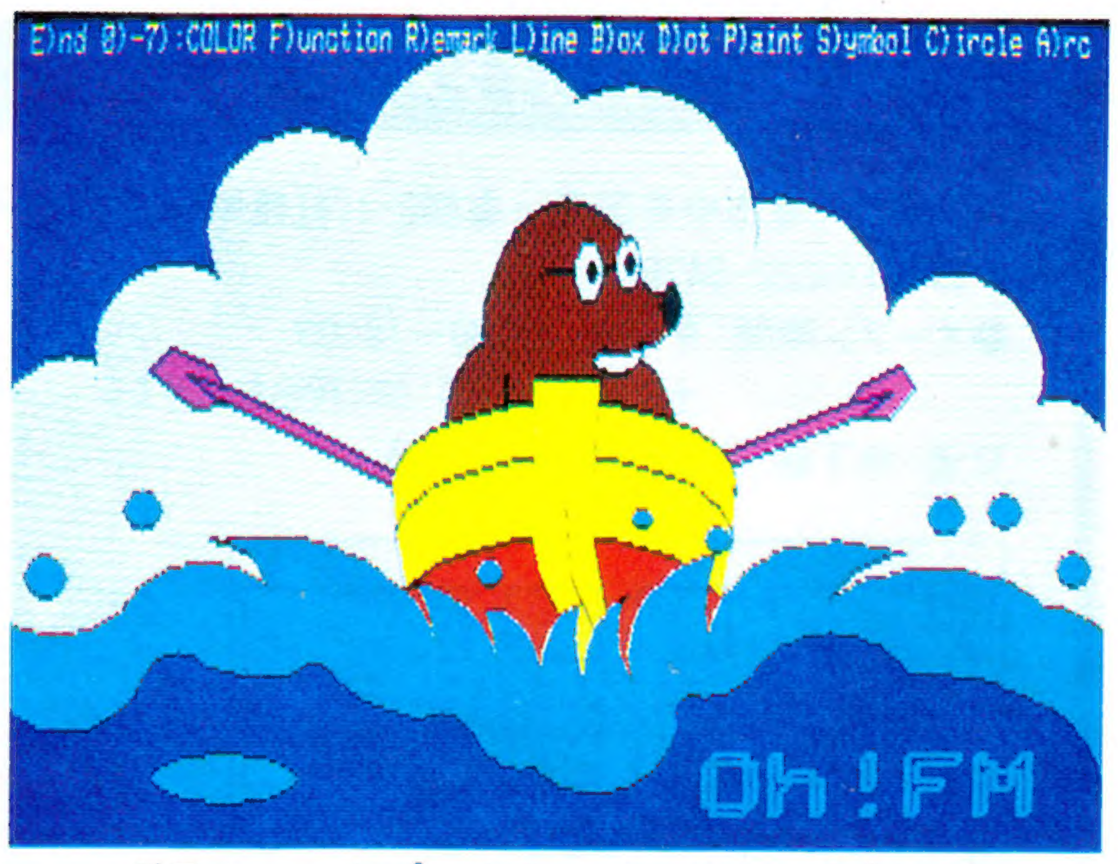
C86 PC98の5インチディスクをFMで読む/青山千秋 …… 53

OS9 OS-9サービスリクエストの使い方/紀尾井太郎 …… 56

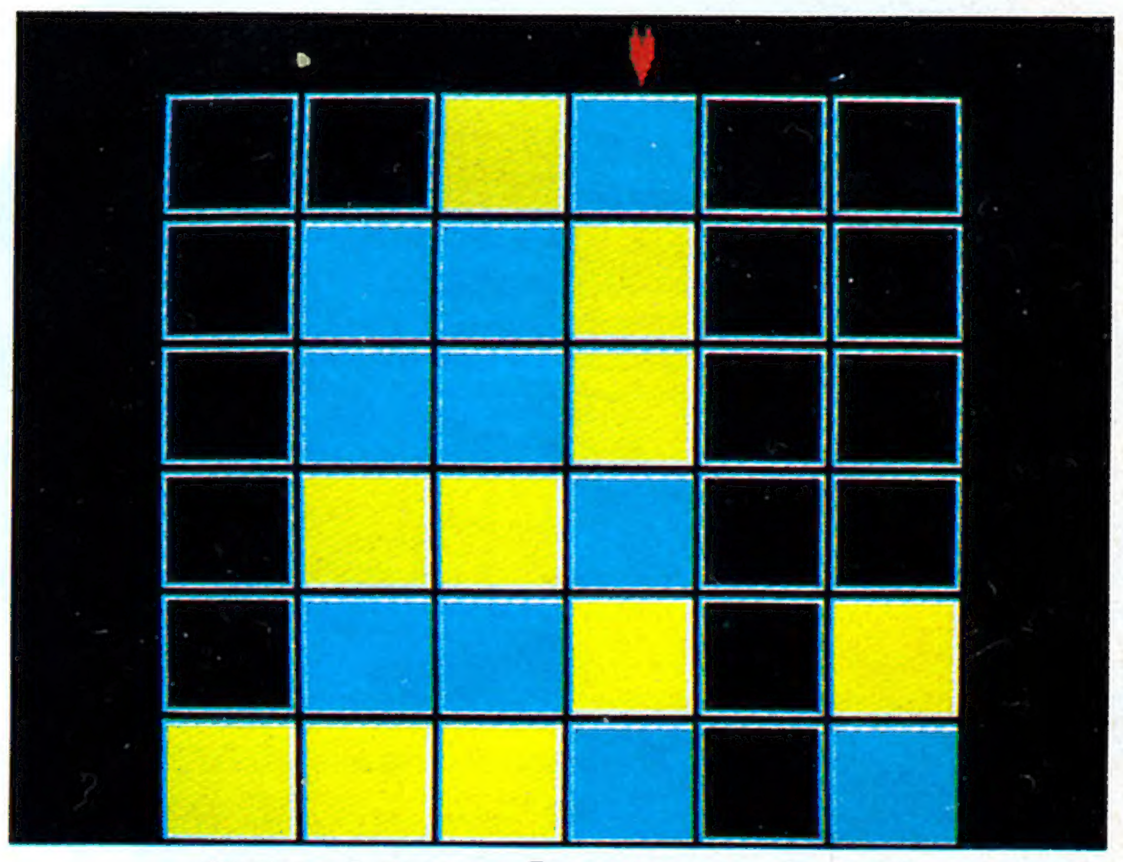
OS9 OS-9逆アセンブラV1.0/Simple Logic …… 61

7 8 11 プリンタポートで遊ぼう
簡易ジョイスティックの製作/傍若無人 …… 123

7 8 11 6809機械語マスタ'84
スタックと特殊なアドレッシング/鶴岡哲明 …… 97



グラフィックプログラムジェネレータ(P.86)



縦積み4目並べ「HAMLET」(P.152)

7 8 グラフィックプログラムジェネレータ/古田 昌……86

はじめてさわるパソコン/小林泰介……109

7 8 11 パソコンの計算精度/阿部 正……91

7 8 11 アマグラマ講座
あなたが作る手作り学習プログラム入門/国重 博……74

7 8 11 ビジネス用BASICプログラムジェネレータ
[6] ソートプログラムジェネレータ/高木征蔵……132

C86 BDOSコール入門/西村義孝……113

SOFT REVIEW……81

ショートプログラム集……149
レーシングゲーム/サブマリンファイト/迷路ゲーム/縦積み４目並べ「HAMLET」/DATA SEARCHER

READERS' AREA……146

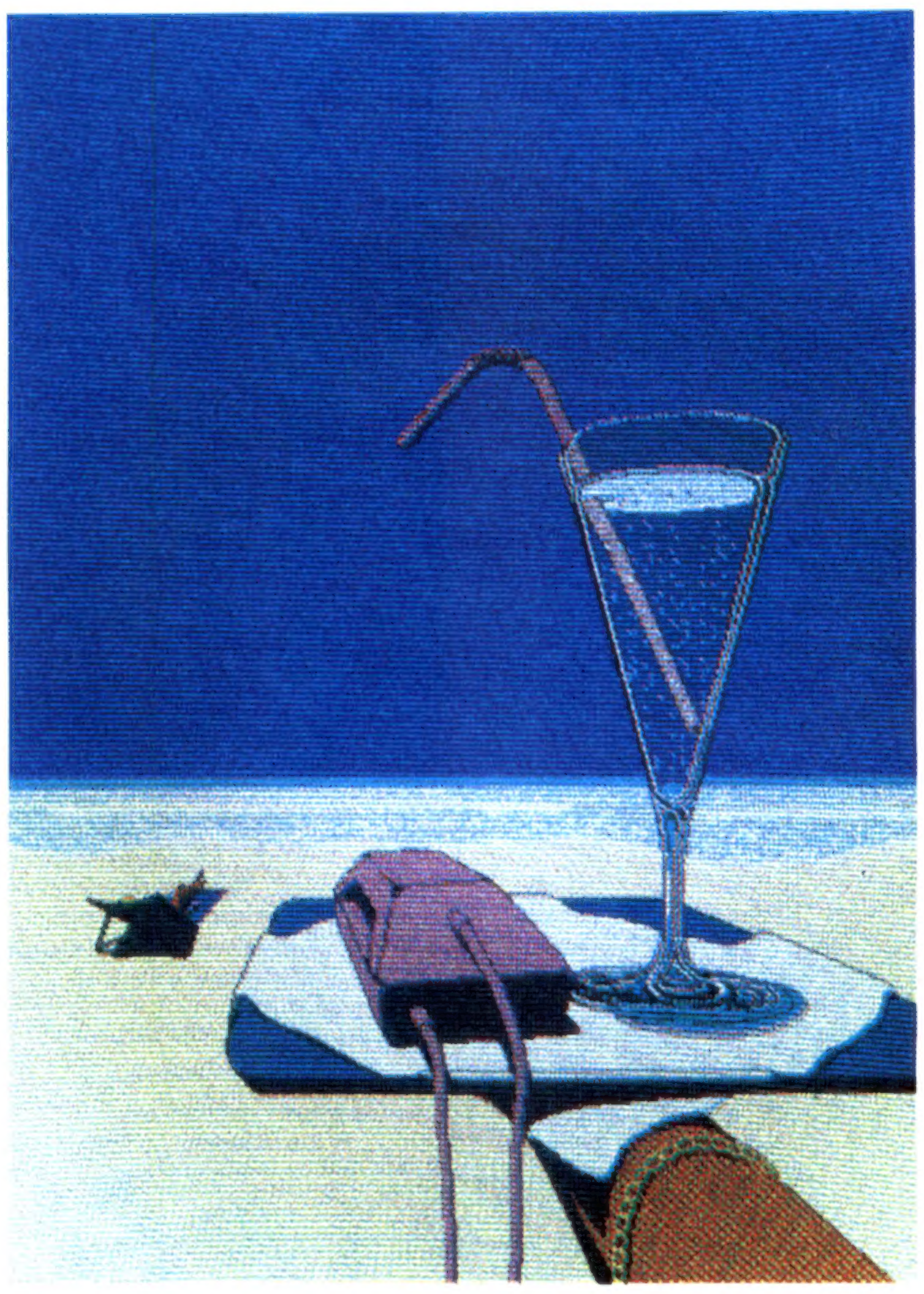
FMぽーと……145

プレゼント情報……145

虫つくろいのページ……145

広告索引……176

EDITORS' ROOM……176



「お嬢様，お電話が入っております。」
Tea Spoon 木下淳博

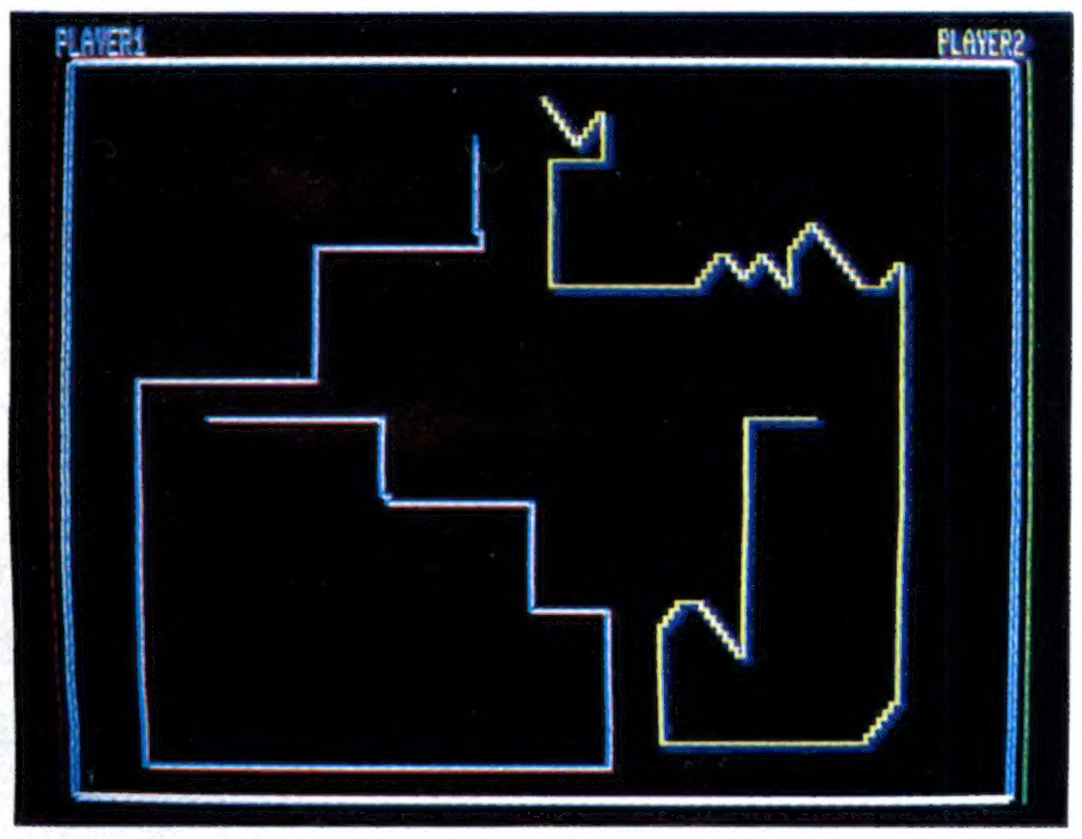
あの太陽は本当に動いているのだろうか。壊れた柱時計を見ているようだ。自分と波だけに時間が流れている。

もうすこし待とう。潮が満ちて，波が足を洗ったら，すべてを忘れてここを出よう。

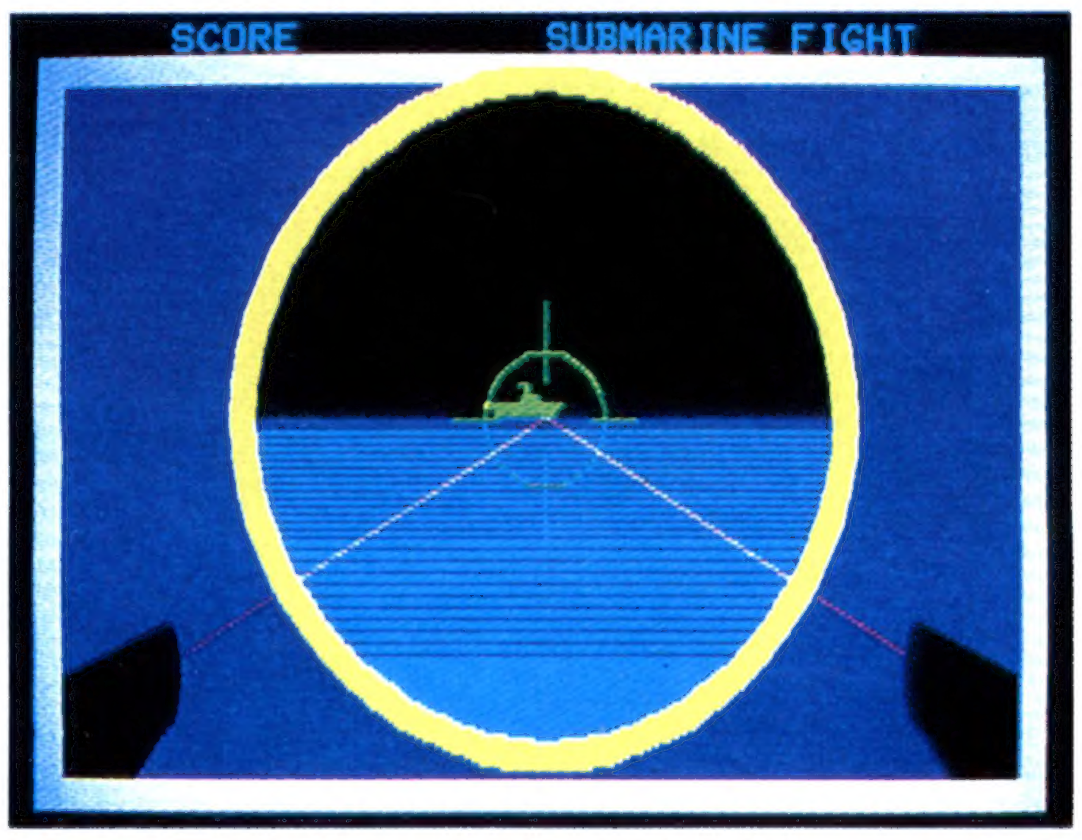
その時であった。

・CP/M, CP/M-86, MP/MはDigital Research社
・MS-DOSはMicrosoft社
・FLEXはTSC社
・UNIXはAT & T Bell Laboratories
・UCSD p-Systemはカリフォルニア大学理事会
・OS-9はMICROWARE社
・SB-80, SB-86はLIFEBOAT ASSOCIATES
・WORDSTAR, MAILMERGE, SPELLSTAR, WORDMASTER, CALCSTAR, DATASTAR, SUPERSORT, INFOSTARはMicro Pro社
の各メーカーの登録商標です。
・MSXマークはマイクロソフト社の商標です。
その他プログラム名，システム名，CPU名は一般に各開発メーカーの登録商標です。
本文中では“TM”，“®”マークは明記していません。
本誌に掲載されたすべてのプログラムは，個人で使用する以外著作権法上，無断複製を禁じられています。

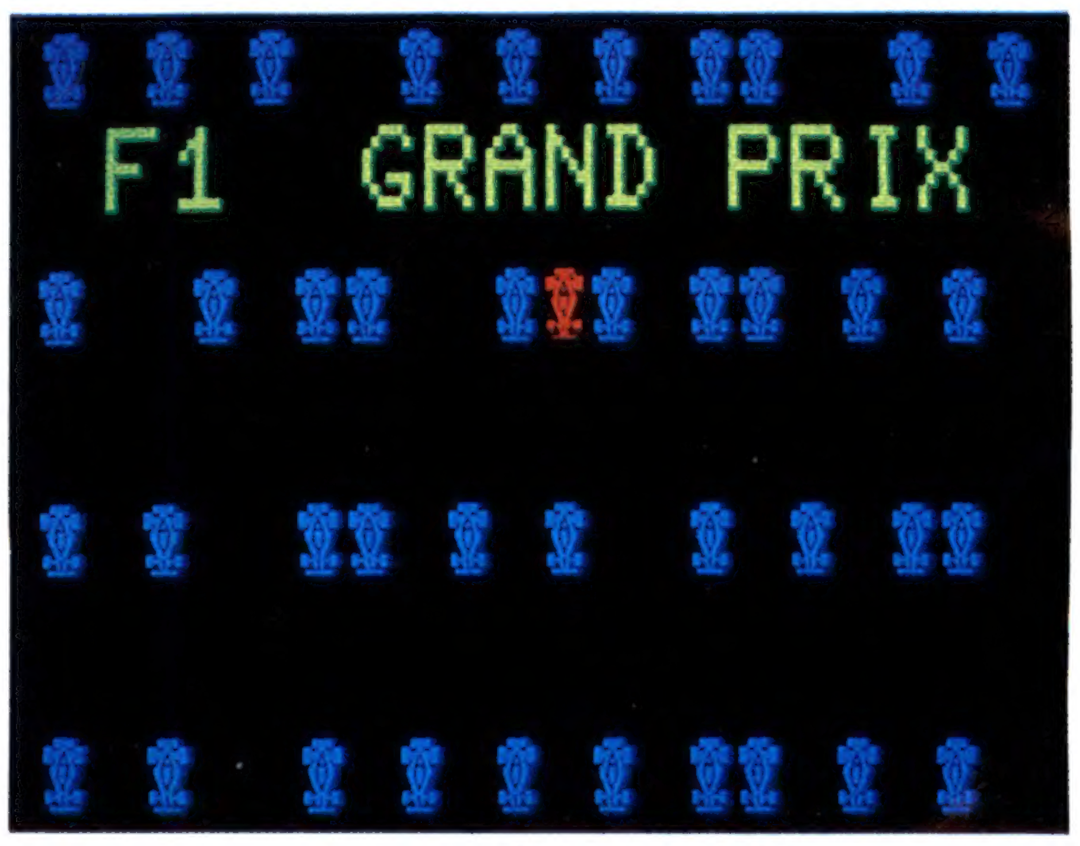
●機種表示について●
7……FM-7, FM-NEW7, FM-77
8……FM-8
11……FM-11/ST, AD, EX, AD2, BS
X……FM-X
C86…CP/M-86の走るFMシリーズ
OS9…OS-9の走るFMシリーズ
FLX…FLEXの走るFMシリーズ



SPEED TRAP(P.130)



サブマリンファイト(P.149)



レーシングゲーム(P.149)

ソフトとソフトをつなぐニュータイプ

FMシリーズ用 "統合化"データベースソフト

レターバンクは,
●顧客管理・文献管理等に利用できます。
●各種の献索・分類業務に最適です。
●各種通知、文書をワープロ機能で作成できます。
●文書の差し込み印刷、各種のハガキ、封筒、タックシールへの印刷ができます。
また、●オールマシン語プログラムのため、すべての処理を高速で行ないます。
●当社のワープロ、データベースと連携できます。
●手頃な価格、コストパフォーマンスは最高です。

＊ご注文、お買い上げの節は、機種名を明示してください。
プリンターは、ドットプリンターでも漢字プリンターでも、ほとんどのものに対応しています。

Letter Bank
レターバンク

メディア	価　格
3.5″FD,5″FD(320KB)	¥33,000
5″FD,8″FD(1MB)	¥38,000

対応機種：FM-7, New7, 77
FM-8
FM-11(BSを除く)

PC-8801ユーザーに朗報。

COMAS-WPN5とエプソンTF-50をセットで販売します。
(日本語ワープロソフト)　(5″2HDディスクドライブ)

高度な操作性,高速性を誇るWPシリーズと、TF-50の大容量,高速性がマッチして、文書の処理能力が大幅にアップします。

セット価格は、22万円
当社パソコン事業部まで、直接ご注文ください。

日本語ワープロソフト
COMAS-WPシリーズ

対応機種：FMシリーズ
(FM-11 BSを除く)
PC-8801, 8801mkII
PC-8001mkII

価　格：メディア,製品レベルにより
¥38,000〜¥75,000

日本語データベースソフト
COMAS-K2シリーズ

対応機種：FMシリーズ
(FM-11 BSを除く)

価　格：	320KB FD	1MB FD
K2E	¥55,000	¥ 85,000
K2G	¥85,000	¥120,000
K2S	¥99,800	¥135,000

給与計算＋日本語データベース
COMAS-K2給与

対応機種：FM-7, New7
FM-8
FM-11 AD2

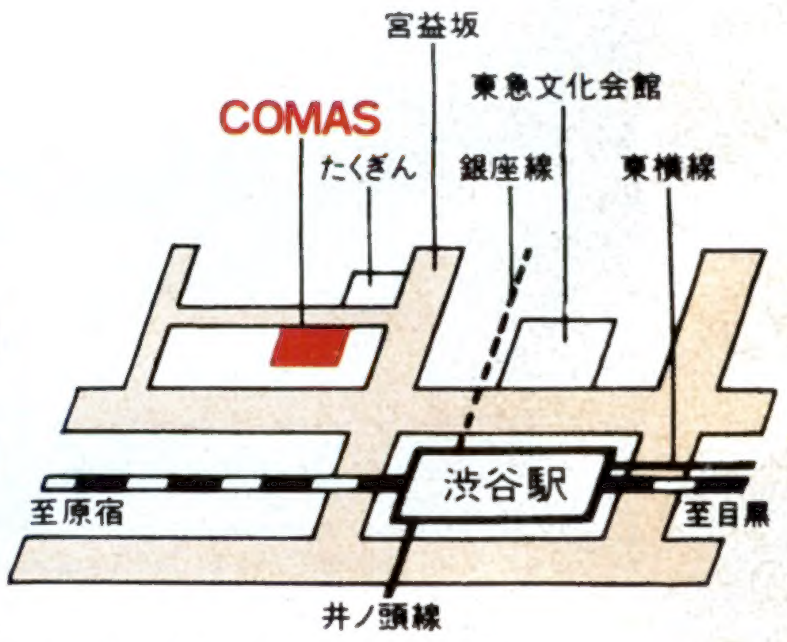
価　格：¥119,800 (5″2D)
¥155,000 (8″2D, 5″2HD)

COMAS-K2活用講座

●毎週水,金曜日●一般,専修,夜間コース　(各コースとも3時間)
●受講料：¥5,000●場　所：当社教室
なお、関西方面では,産業技術研究所(TEL.075-981-4080)で開設しております。詳細は上記までお問い合わせ下さい。

コピー商品にご注意ください。
当社の製品には当社宛返送用の〔COMASソフト製品登録カード〕が付属していますので、
製品お買い上げの節には、かならずこの点をご確認のうえ当社宛ご送付いただくようお願い致します。

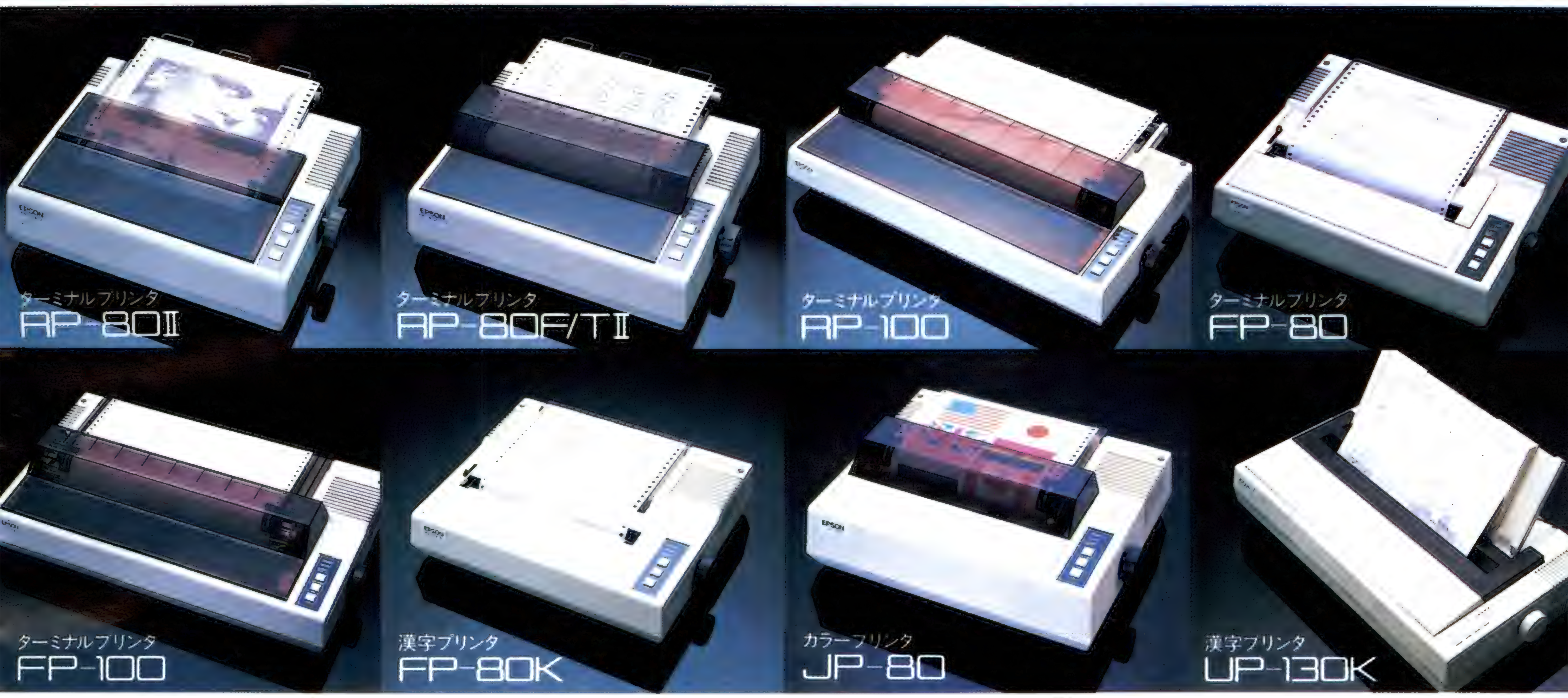
Computer Application Service
COMAS
株式会社コマス/パソコン事業部
〒150東京都渋谷区渋谷1-14-6(平野屋ビル) TEL.(03)407-8893(代)



EPSON

ペリフェラルはエプソン

ジャストフィット! FMシリーズ対応機。



RP-80II ●ハイ・コストパフォーマンスを誇るターミナルプリンタ¥79,000
RP-80F/TII ●用紙を選ばないハイ・コストパフォーマンスタイプ ¥89,000
RP-100 ●RPシリーズのハイ・コストパフォーマンスを受け継ぐ136桁プリンタ¥134,000
FP-80 ●ビットイメージ、プロポーショナル印字など多機能で高速 ¥149,800
FP-100 ●FP-80の機能に加え、136桁印字を実現 ¥189,800
FP-80K ●FP-80の多機能を搭載した漢字プリンタ ¥189,800
JP-80 ●7色のカラー印字が可能な多機能・高速ドットマトリクス・プリンタ ¥158,000
UP-130K ●高速、高印字品質の漢字プリンタ¥290,000(第1水準)・¥310,000(第2水準)

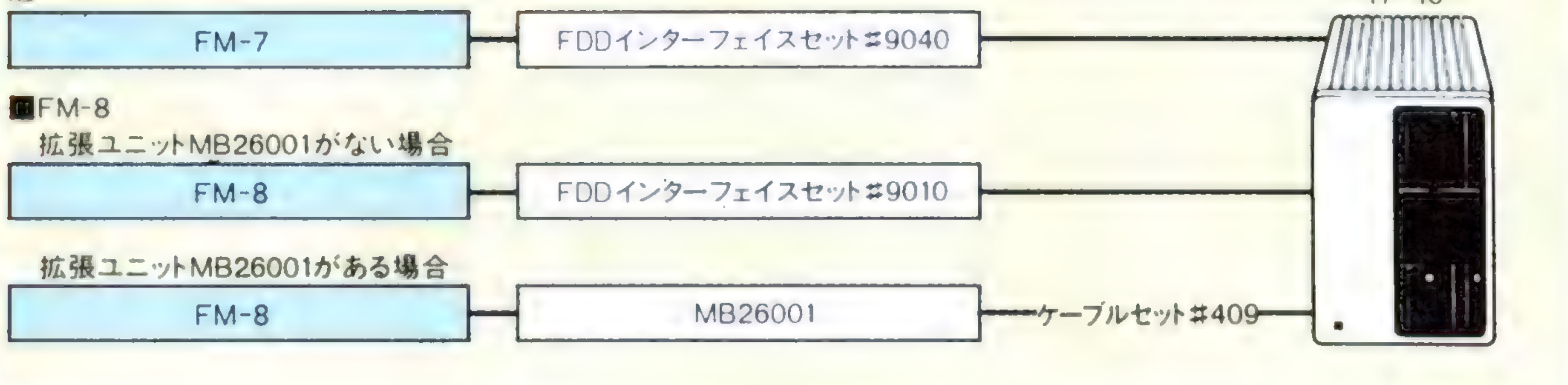
●FMシリーズ対応表

	RP-80II RP-80F/TII RP-100 標準モード	FP-80、 FP-100 標準機	FP-80K 標準機	JP-80	UP-130K 標準機
FM-7 I/Fケーブル#8224(¥8,000)により接続	○ 可 推奨	○ 可 推奨	○ 可	○ 可(注1)	○ 可
FM-8 I/Fケーブル#8224(¥8,000)により接続	○ 可 推奨	○ 可 推奨	○ 可	○ 可(注1)	○ 可
FM-11ST/11AD/11EX/11BS I/Fケーブル#8225(¥8,800)により接続	○ 可 推奨	○ 可 推奨	○ 可	○ 可(注1)	

キャラクタ・ジェネレータ/O:文字コード表一致　△:グラフィックキャラクタのみ不一致　ハードコピー/可:CRTハードコピー可能　(注1):ユーティリティによりCRTのカラーハードコピー可能

TF-10 ●ハイ・コストパフォーマンスと信頼性を誇るターミナルフロッピー
FMシリーズ 専用機¥125,800(デュアル)¥85,000(シングル)

●接続方法



●エプソンのプリンタは、ESC/P™(EPSON STANDARD CODE for PRINTER)のもとにターミナルプリンタ・コントロールコード体系の世界統一規格を提唱し製品開発されています。

エプソン株式会社 ●本社/〒399-07長野県塩尻市広丘原新田80☎0263-52-2552代 エプソン販売株式会社 ●本社/〒160東京都新宿区西新宿2-4-1新宿NSビル内私書箱6109号☎03-348-6801代・中央営業所/〒101東京都千代田区神田須田町2-21サナダビル7階☎03-258-4841・大阪支店/〒530大阪市北区西天満4-14-3住友生命御堂筋ビル11階☎06-365-5071・大阪南営業所/〒556大阪市浪速区日本橋5-4-20エスタビル4階☎06-632-3353・札幌営業所/〒060札幌市中央区北一条西2丁目札幌時計台ビル6階☎011-222-2821・秋田営業所/〒010秋田市中通2丁目176秋田東京海上ビル7階☎0188-32-4002・仙台営業所/〒980仙台市本町1-12-12山万ビル6階☎0222-63-3691・長野営業所/〒390松本市深志2-2-9加納ビル2階☎0263-36-7251・新潟営業所/〒950新潟市東大通1-2-25北越第1ビル6階☎0252-43-8515・名古屋営業所/〒460名古屋市中区新栄町2-13栄第一生命ビル9階☎052-962-7001・金沢営業所/〒920金沢市彦三町2-3-13明治生命金沢彦三ビル4階☎0762-62-3216・広島営業所/〒730広島市中区鐵町3-1第3山県ビル4階☎082-223-2811・福岡営業所/〒812福岡市博多区博多駅前3-4-8郵政互助会博多ビル4階☎092-471-0761・鹿児島営業所/〒892鹿児島市加治屋町18-8三井生命鹿児島ビル4階☎0992-25-7717・エプソンサービスセンター/〒101東京都千代田区神田須田町2-21サナダビル6階☎03-258-4881●詳しい資料のご請求は、はがきに住所、氏名、年令、職業、製品名をお書きの上、もよりのエプソン販売までお申込みください。

ペリフェラル
資料請求券
OH!/FM

死後、
お役に立ちます。
天使たちの共通一次
地獄の練習問題
©
Humming Bird Adventure#0
奈落の底から、新発売。
「本試験は、完全な実技試験である。
受験者は、一介の亡者となって、
様々な地獄の試練に耐え抜き、
再び自力で天上界に戻ってこなければならない。
脱出経路や方法は、すべて自分で発見せねばならない。」
天使庁上級天使選考試験〈受験要項〉より
アドベンチャーゲーム
富士通FM-7・8・new7用5インチディスク版 FM-77用3.5インチディスク版
¥6,800
Humming Bird Soft™
㈱エム・エー・シー コンピュータ事業部
〒530 大阪市北区曽根崎2丁目2番15号 ☎06(315)0541㈹

僕らのパソコンと相性ぴったりのプリンター、ブラザーから「白」と「黒」で誕生！ホワイトボディはFMシリーズ対応のHR-5。ブラックボディはPCシリーズ・MSX対応のHR-5X。どちらも乾電池駆動の超コンパクトサイズ。美しく、静かな印字が自慢です——。僕らのパソコン・パートナーは、これに決まり。

■主な特長

- 超軽量1.6kg、乾電池駆動で機動性抜群。
- 熱転写／サーマル方式だから、印字音はきわめて静か。普通紙にも印字可能。
- ハードコピーまたはアプリケーションソフトにて漢字印字可能。
- 豊富な機能を低価格で実現。ほとんどのパーソナル・コンピューターと接続可能。

■主な仕様

印字方式：熱転写／感熱方式（ノンインパクト）
印字ヘッド：9ピンサーマルヘッド
文字種類：〈HR-5〉JIS159種＋グラフィック63種
〈HR-5X〉JIS159種＋グラフィック56種＋ひらがな64種（ACモード） 256種（Bモード）
文字構成：〈HR-5〉通常文字—9×9ドットマトリックス（ハーフドット4）
グラフィック—6×8ドットマトリックス
〈HR-5X〉通常文字—9×7ドットマトリックス
グラフィック—8×8ドットマトリックス
印字速度：約30cps
電源：2電源方式DC6V＋10%～−20%
乾電池単1 4本またはACアダプター
外形寸法：303（幅）×174（奥行）×65（高さ）mm

このHR-5&5Xについて、詳しく知りたい方は………ハガキに資料請求券を貼り、お手持ちのパソコン機種、使用方法（ゲーム、ビジネス…etc）、氏名・年令・電話番号をご記入の上お送り下さい。

資料請求券
Oh!FM
84-9

ブラザー販売株式会社 情報機器事業部

東京 〒104 東京都中央区京橋3-3-8 TEL(03)274-6911
名古屋 〒460 名古屋市中区大須3-46-15 TEL(052)263-5811
大阪 〒542 大阪市南区心斎橋筋1-1 TEL(06)251-7265

生きたカードで未来を占う！

動く！スーパー・タロット

近日発売!!

●今までのパソコン占いソフトの常識をやぶる!!

●100%に近い的中率をほこるタロットカードを超グラフィックで再現!!

FM-7
FM-NEW7
FM-77

- ●3.5/5インチディスク版（2枚組）
- ●オール・マシン語による高速グラフィック
- ●漢字表示（漢ROM不用）

募集中!!……オリジナルソフト（ゲーム・ビジネス・ユーティリティ）
★詳細条件はお問合せ下さい。

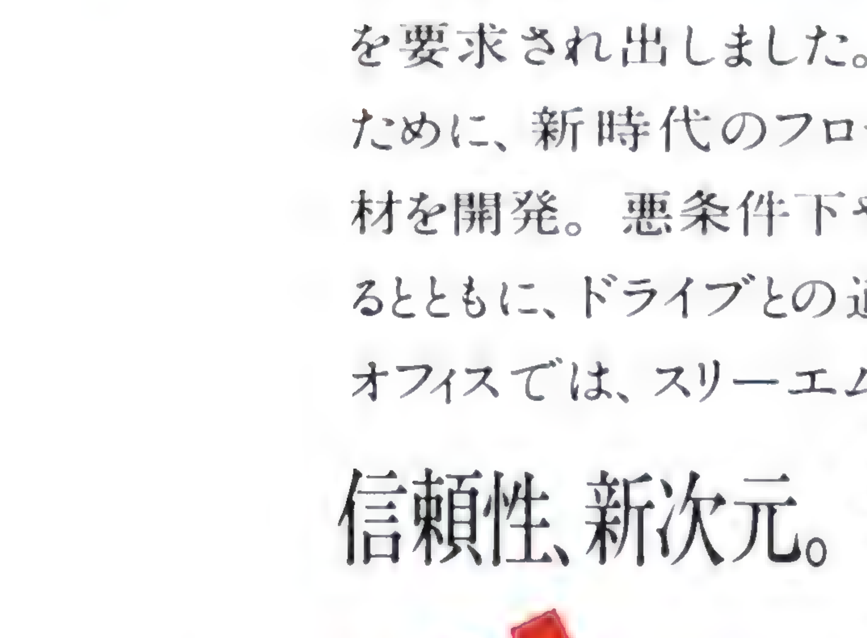
通販

●通信販売は住所・氏名・電話番号・機種名・商品名を明記の上、右記の宛先まで現金書留でお送り下さい。
尚、通販ユーザー激増につき、今月より送料ナシ!!

Policy 株式会社ポリシー
〒541 大阪市東区南久宝寺町2-71-2 船場双葉プラザ
TEL 06(261)4334(代)

直径わずか5.25インチや8インチのフロッピーディスクの中に、広大な情報宇宙が広がっています。その情報宇宙も、従来の倍密度タイプに比べ3倍以上の1.6メガバイトもの容量をもつMD/2HD(5.25インチ高密度タイプ)の誕生によって、加速度的に規模を拡大し、さらに信頼性の向上を要求され出しました。スリーエムは、情報宇宙を存分に活用し楽しむために、新時代のフロッピーディスクをめざし、新磁性体、新構造、新素材を開発。悪条件下や酷使に耐え抜く耐熱・耐低温・耐湿性を完成するとともに、ドライブとの適合性の飛躍的な向上を実現しました。家庭やオフィスでは、スリーエム フロッピーディスク。新次元の信頼性を提供します。

信頼性、新次元。



スリーエム
フロッピーディスク
新発売

住友スリーエム株式会社 3M
磁気製品事業部
本社 158 東京都世田谷区玉川台2-33-1 ☎(03)709-8526

東京支店 ☎(03)403-1111
横浜支店 ☎(045)312-5521
名古屋支店 ☎(052)332-2411
大阪支店 ☎(06)305-3131
福岡支店 ☎(092)531-4333
札幌営業所 ☎(011)644-7411
仙台営業所 ☎(0222)61-2811
広島営業所 ☎(082)247-2200
沖縄連絡事務所 ☎(0988)77-8799

3M

技術解析シリーズによって培われた本書は体系的で

これまでの入門書とは一線を引いて

FM-7/NEW7/77
マシン語入門マニュアル

解析書の秀和が中・上級者向けに贈る『入門マニュアルシリーズ』第1弾。FM-7のBASICに限界を感じた方に、マシン語を基礎から指導。ゲーム制作などのマシン語プログラミングも自由自在。FM-7/NEW7/77を完全に使いこなすユーザ必読の一冊。

B5判　全280頁　定価2,200円　送料300円

For the systems engineer

秀和システムトレーディング株式会社
〒107 東京都港区南青山2-19-5　関原ビル
☎03(470)4941(代)　ファクシミリ405-7538
秀和システム

技術提携/㈱ フォーマット・ジャパン

●お問合わせはお近くのマイコンショップ、書店へ。通信販売も致します。ご注文の際は送料を添えて現金書留または、銀行振込でお願いします。●振込先/住友銀行麻布支店・普通預金No.127404ご入金の通知があり次第、現品を発送致します。●振込の場合はあらかじめ住所、郵便番号、氏名、ご注文の品名、個数をハガキにてお知らせ下さい。

資料請求券
Oh/FM '84.9

実践マニュアルシリーズ

PC-8801・PC-8801mkⅡ
UTILITYPROGRAM
ユーティリティ・プログラム応用実例集

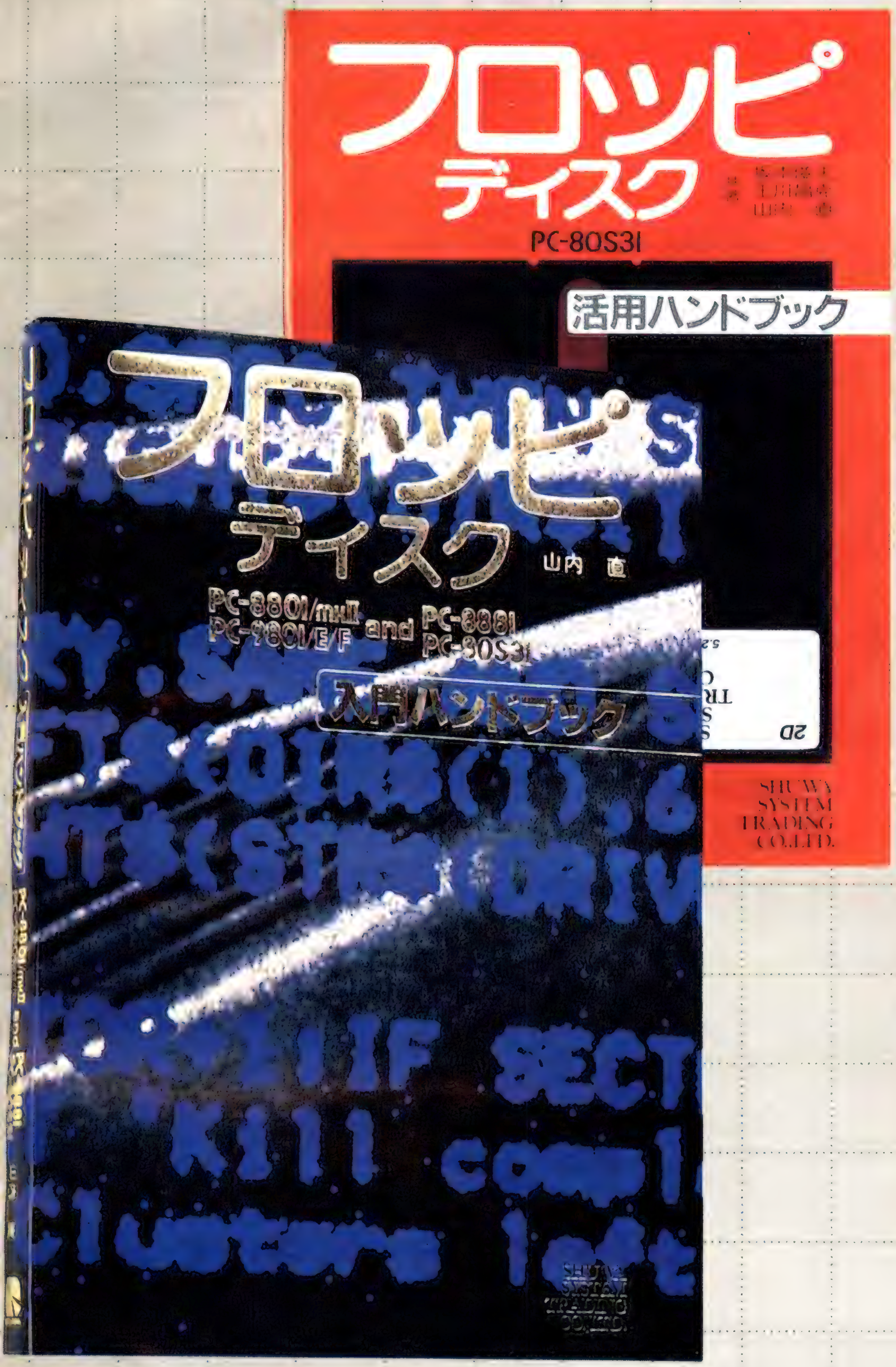
FM-7/NEW7/77
UTILITYPROGRAM
ユーティリティ・プログラム応用実例集

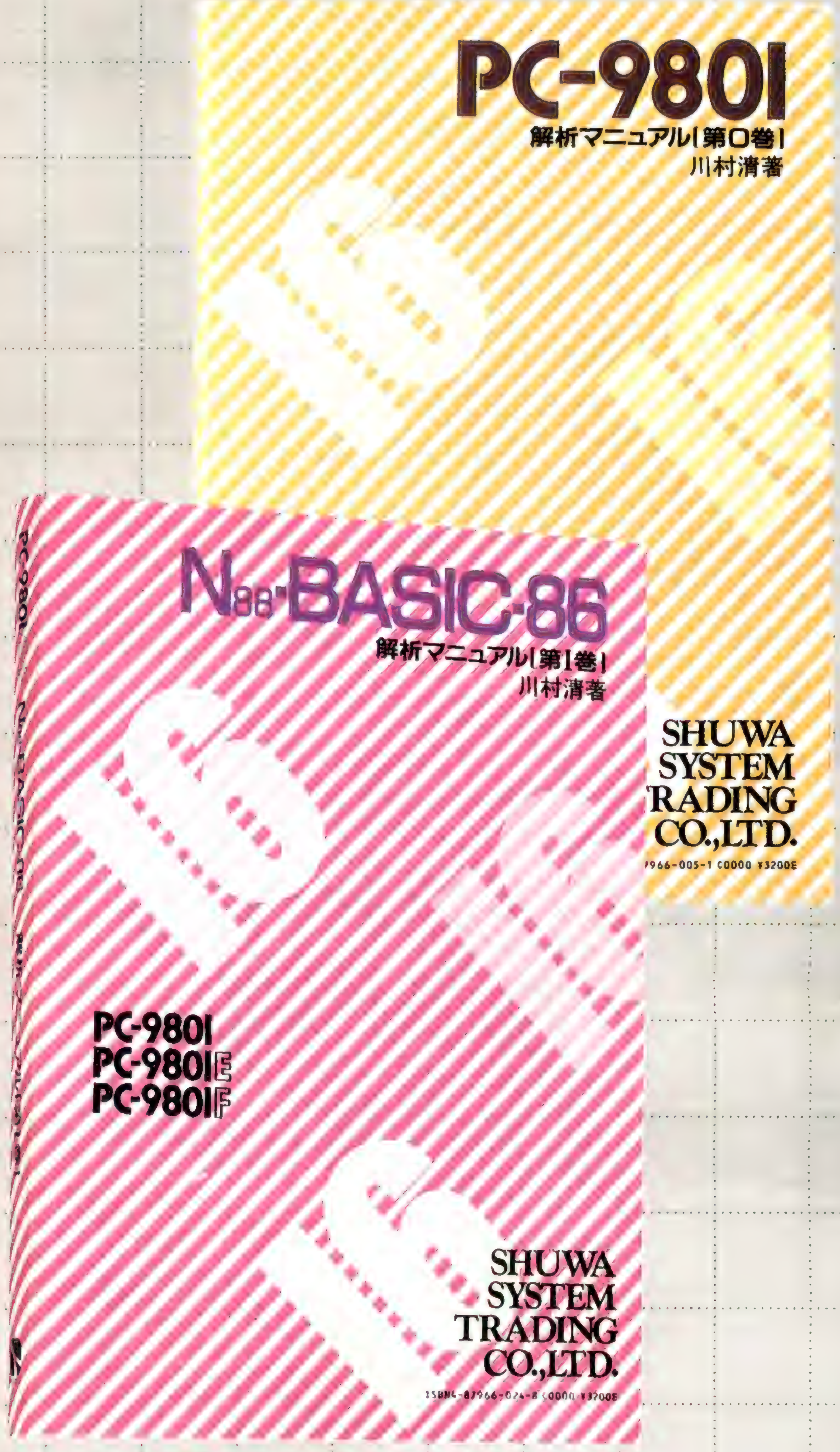
PCユーザとFMユーザの皆様から寄せられた長い間の熱望にお応えして——秀和システムトレーディングのスタッフが技術の一端を提供。プログラミングテクニック集。

A5判　全450頁　定価2,500円　送料300円

A5判　全450頁　定価2,500円　送料300円

わかりやすく啓発的で センスのよい一冊である





入門マニュアルシリーズ

フロッピディスク入門ハンドブック

PC-8801/mkII PC-9801/E/F and PC-8881 PC-80S31

Personal Computer System の能力を100%引き出すために!!
ディスクを実用的に扱うための基礎的な知識一概念から、高度なアルゴリズムに至るまで、豊富な図表と実例を使って活用ハンドブックの著者が徹底的に述べる第2作。

B5判　全240頁　定価2,500円　送料300円

解析マニュアルシリーズ

N88-BASIC-86 解析マニュアル〔第1巻〕
BIOS-9801からN88-BASIC(86)へ
PC-9801/E/F 対応

N88-BASIC(86)を最高のサブルーチンパッケージとして活用したい——長大な月日を要して、このような着想の実現化をはかりました。**PC-9801** 技術解析シリーズを通してプログラミングの省力化を提唱する。

A5判　全370頁　定価3,200円　送料300円

解析マニュアルシリーズ

FM-7・8 OS-9 Level 1 解析マニュアル1

UNIX ライクオペレーティングシステム**OS-9**の全貌を……いま。
8ビット機唯一のUNIX ライクOS、OS-9全体をカーネル中心に解析し、プログラム実行の手順を豊富なサンプルプログラムと共に詳細に解説する。
FM-77/NEW7に対応。

A5判　全330頁　定価2,800円　送料300円

近刊予定

PC-9801/E/F 対応
ユーザ待望の続刊!

インタフェース編
ディスク編
グラフィクス編
モニタ編

予価3,200円

SONY

日本全国のパソコンファンへ。

①0.37㎜ファインピッチ②2000文字対応③RGB入力

端子 ⑤ 傾斜画面(スラントパネル) ⑥テレビ番組、ビデオも見られる。

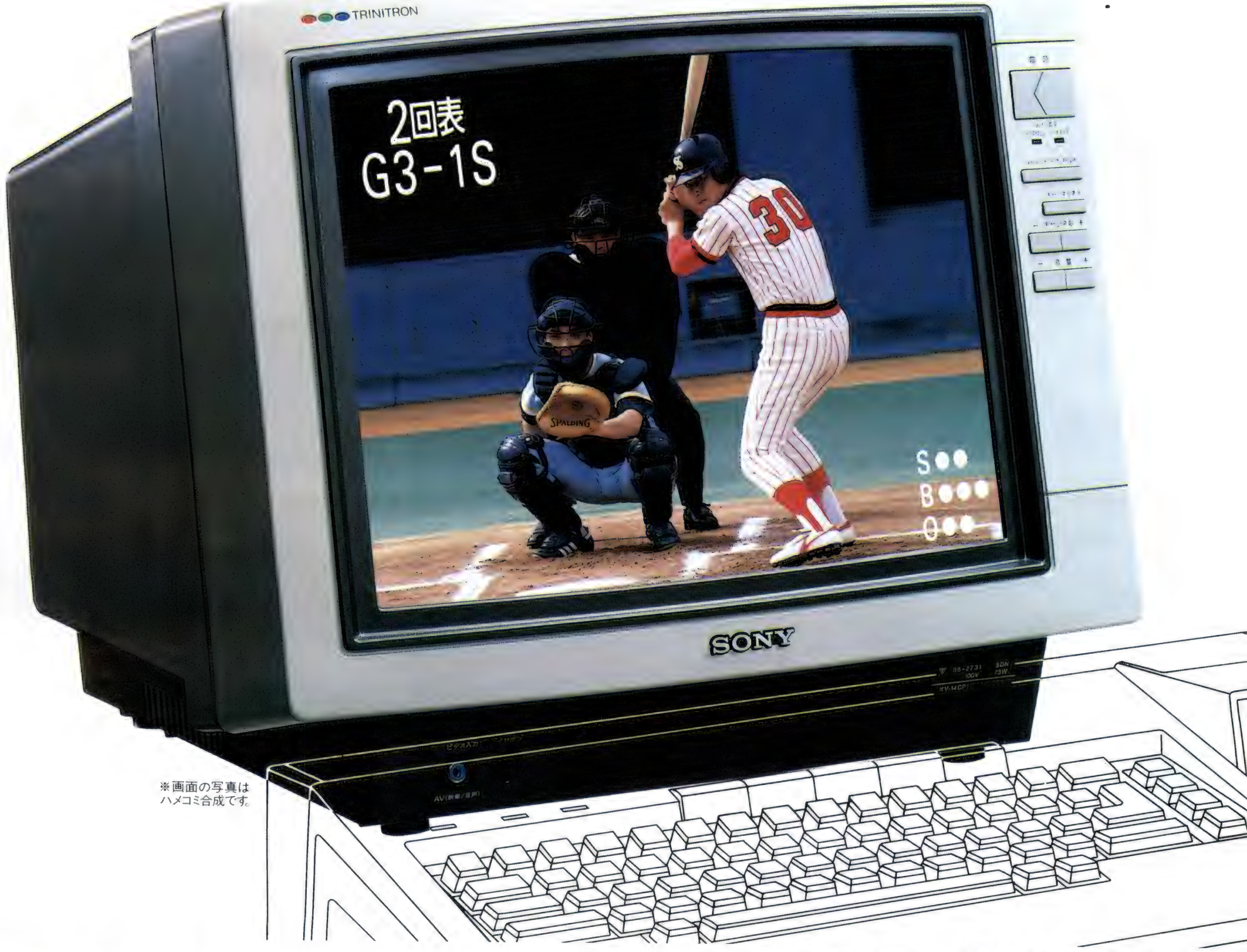
カラフル。鮮明。目にやさしい、と人気のブラックトリニトロンにファインピッチを加えた、カラーディスプレイが出ました。KV-14CP1。これは、びっくりするほどの高画質が自慢です。従来の14G1のCRTに較べ約60%も密度が高い0.37㎜ファインピッチで、いままで味うことのできなかった映像美を楽しめます。パソコン用カラーディスプレイとしての素晴しさは、これだけにとどまりません。2000文字を鮮やかに表示する。アナログ(21ピン)/デジタル(8ピン)RGB入力端子を装備している。前面を10°傾けたスラントパネルを採用している。いままでのパソコン用ディスプレイとは、はっきり違う

ソニーならではの特長を備えています。コンピュータグラフィックスをより美しく、プログラミングをよりやさしく、ゲームをより楽しくするのが、KV-14CP1です。さらに、映像ソースを選ばない豊富な入出力端子も使う人にうれしいところ。オーディオ、ビデオに対応する前面AV Hit/ビデオ入力端子。モニター出力端子。テレビ出力端子。音声多重出力端子。と、実に様々な楽しみに対応できる設計。オーディオ、ビデオ、ビデオディスク、SHF放送、文字多重、音声多重などを美しく映し出します。「チャンネル」「音量」「チューニング」「RGB」「ビデオ」などを管面表示するので、気持ちよく使

©by FELIX THE CAT PRODUCTIONS, INC. by King Features Syndicate, INC.

（上手に使って上手に節電

あなたのためのテレビです。



※画面の写真は
ハメコミ合成です。

④前面AV Hit／ビデオ入力

ブラックトリニトロン、新発売。

えます。最後に、パソコンに疲れたら、テレビ放送をすぐ楽しめるのも大きな特長。パソコンディスプレイとして十分な美しさを誇るのだから、テレビ放送を映せば、美しすぎるほど美しい。夏の甲子園の芝生なんか、ハッとするほどの鮮やかさで迫ってくること、うけあいです。ブラックトリニトロン＋ファインピッチ。パソコンをこれほど美しく見せるテレビがあったでしょうか。次の時代の美しさを映すテレビ。KV-14CP1。

BLACK TRINITRON

ソニートリニトロン カラーテレビ
14型 KV-14CP1 ¥99,800

別売AV Hit端子用オーディオ・ビデオケーブル ●VMC-610S ¥1,500（ステレオ用） ●VMC-611MS ¥1,500（モノラル用） ●VMC-612MS ¥1,700（モノラル用）

●カタログ請求は、住所・氏名・年令・職業・電話番号・機種名を明記の上、〒108東京都高輪局区内ソニー㈱カタログ係へハガキで、お申し込みください。

資料請求券を
ハガキにお貼り
下さい。FM

Falcom

FM-ユーザーに朗報！

新作ニュース新作ニュース新作ニュース新作ニュース新作ニュース新作ニュース新作ニュース新作ニュース新作ニュース新作ニュース新作ニュース新作

前代未聞麻薬的爽快遊戯

Dragon Slayer ドラゴンスレイヤー

ROLE PLAYING ロールプレイングゲーム と ACTION アクションゲーム が CROSS OVER クロスオーバー しました。

10月発売予定！ FMユーザーの方々、お楽しみに〜！

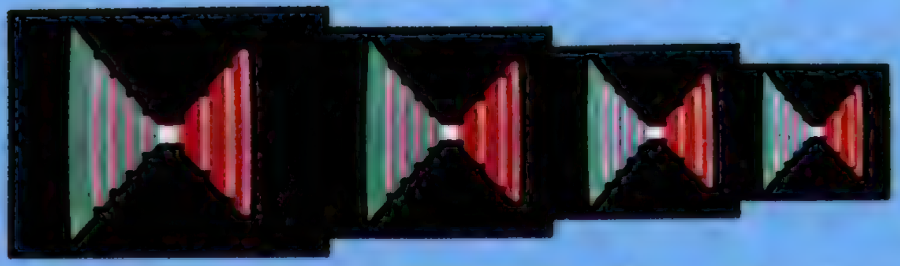
プレイヤー次第で、無限に拡がる桃源郷。ロールプレイングをアクションゲームのフィーリングでプレイできる！全方向スクロールアドベンチャー。世界に類を見ない超ユニークなゲーム。勿論！オールマシン語です。
乞御期待！

今宵、貴方を異次元紀行へ

ESCAPE FROM TWILIGHT ZONE

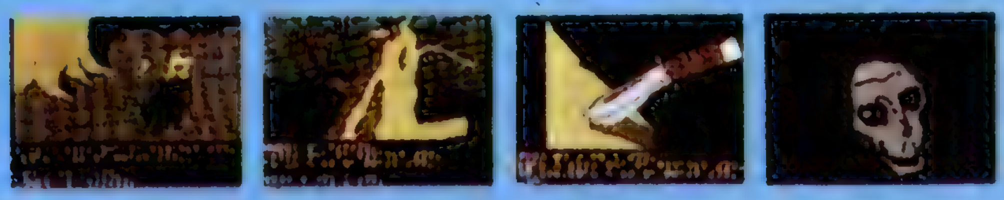
全てにおいて、今迄の常識を超えた革命的アドベンチャーゲームです！ 充実内容で¥3,800という低価格の実現など、日本ファルコムでは、FM-7をカセットベースでお使いの方に本格的アドベンチャーゲームを知って頂く為、開発致しました。

ロードが終ると、電話ベルが突然鳴り出しゲームが始まる。ゲームの流れのポイントごとに効果がバッチリと使われていて、冒険の臨場感も最高の物が楽しめます。



時空の迷宮内の画面表示は平均1秒！他の場面でもグラフィックは、オールマシン語、高速描写！電子冒険小説ならではのスムーズな、ゲーム展開が楽しめます。

テープのロードは最初の1回、ゲーム中に何度もロードを重ね、冒険心を損なうことはありません！ 又、一つの場面を解決できなければ他へ進めないと云う制約が少ない為、場面間を自由に移動できストーリーの展開に無理がありません。



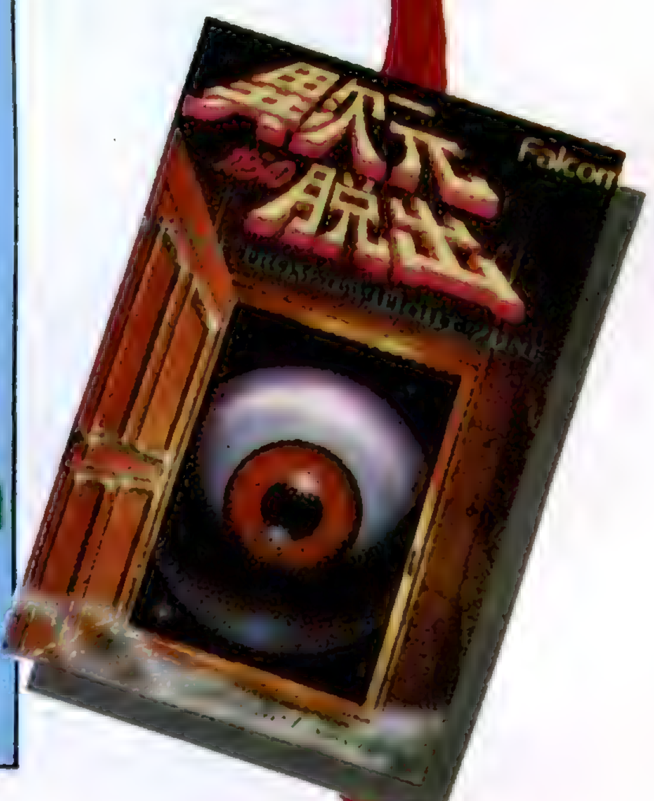
異次元の森の中だけで数十場面。次元の迷宮エリア数が、なんと数百にも及ぶ状大さ！カセット1本の中にこれだけの壮大な世界が拡がっていることに驚かされるでしょう！ これぞアドベンチャーゲームの真髄です！

恐怖の時空移動、超次元紀行

異次元からの脱出

FM-7 カセット¥3,800 5″ディスク¥5,800

FM-77用3.5インチ版¥5,800新登場 (送料〒200)



劇的痛快ソフトウェアー

女子大生プライベート

パズルを解きつつ女の子を脱がして行く、パズル型思考ゲーム。パズルといってもかなりの難しさ、あせる心を迎えつつリアルな彼女達の姿を堪能して下さい。

FM-7 ディスク¥6,200(〒300) カセット¥4,200(〒300)

FM-77用3.5インチ版¥6,200(〒300)新登場！

パソコンが貴方の人生占います〜。

スーパー ホロスコープ漢字版

スーパーホロスコープ漢字版は、FM-7用に開発された漢字対応の西洋占星術システムです。個人はもとより、店頭、学園祭等でのデモンストレーションなどにも、相性診断が出来るので効果抜群です。

FM-7用(送料各 ¥300) カセット ¥6,800
ディスク5″ ¥7,800

FM-77用 3.5インチ版 ¥7,800(〒300) 新登場！

富士通	MB27402, MB27403 MB27406, MB27407
EPSON	FP-80シリーズ, RP-80シリーズ MP-80シリーズ(TYPE・Iは不可)
LOGITEC	KP-3000
スター精密	DPX510F
ブラザー工業	M1009

※上記機種及び、上記機種に準ずるプリンターで使用出来ます。

掲載されているソフトの通信販売も行なっております。注文は、品名・機種名・住所・氏名・電話番号を明記の上、現金書留または、銀行振込でお願いします。銀行振込の場合は、電話又はハガキで商品名を連絡して下さい。 第一勧業銀行立川支店普通264-1436219

日本ファルコム(株) 〒190 東京都立川市曙町1-19-3 10:30〜8:00 水曜日定休 ☎0425(27)4121(代)

フルグラフィックス70画面
メフィウスを凌ぐ一超大作！

暗黒星雲…それは、生命の誕生する分子雲の集合体である。

スターアーサー伝説 II 暗黒星雲

- ★ストーリーは佳境へ 解き明かされるジャミルの正体と、スターアーサーの過去
- ★高速3-Dルーチンを駆使したリアルタイムスペースファイトシーン
- ★本邦初、恐怖の4分割画面（部屋の4方向が同時に表示され、そこで何かが…）
- ★伝説の剣（レイソード）の秘密を暴くべく暗黒星雲へ突入！
- ★3-Dリアルタイム戦闘用修業プログラム付き、リアルタイムのみ独立して楽しむ事も可能です。

絶賛発売中！

FM-7/NEW7/77

テープ版3本組 ¥4,800 5FD版 ¥6,800

ご存じT&E SOFTオリジナルの超高速中間色ペイントルーチンによる大迫力画面、そして圧倒のサウンド。対話モードはもちろんの事、2種類の高速3次元立体図形処理及びビュールーチン（新開発）等様々な画期的手法を駆使。前作"惑星メフィウス"を凌ぐ壮大なスペースファンタジーをお楽しみ下さい。もちろん本編の主人公スターアーサーはあなたです。

新開発
PC-8801用高速中間色ペイントルーチン使用
PC-8801/mkII
5FD版
¥6,800

大好評発売中！

PC-6601
PC-6001mkII
テープ版3本組
¥4,800

スターアーサー伝説 I 惑星メフィウス

	テープ版（3本組）¥4,800	5FD版 ¥6,800
FM-7/NEW7/77	○	○
PC-8801/mkII	○	○
PC-6001mkII/6601	○	
X1/C	○	
S1	○	

FM-77用3,5FD版 新発売!! ¥6,800

3-Dゴルフシミュレーション

ゴルフシミュレーションでは、初めて3次元立体図形処理を使用した本格ゴルフシミュレーションです。毎回のショット位置からの立体視野画面を表示し、その中をボールが飛んでゆきます。（無限に近い視野画面が得られます。）視野画面の作成、ボールの弾道等、すべて3次元計算によって行なっています。また、フェアウェイ・ラフ等の反発係数及び角度、風速、方向、クラブによる打上角度等、ゴルフに必要なほとんどすべての要因を、この計算式に組み込み、実際のゴルフをほぼ完璧にシミュレートします。美しい海岸コース（18ホール）とリアルなサウンドをお楽しみください。

説明書、全コースレイアウト付。

FM-7/8（FD版）超高速化 数秒で次のショットが可能！

	テープ版 ¥4,000	5FD版 ¥6,800
FM-7/8	○	○
X1/C	○	
PC8801/mkII	○	○
PC-9801/E		○
PASOPIA7	○	
S1	○	

T&Eマガジン No.3

ゲームフリークとゲームデザイナーの接点を作ります。

- ●スーパーグラフィックエディター・コンクール発表
- ●コスモミューター開発裏話
- ●ピラミッドII…上達テクニック
- ●連載マンガ…スターアーサー伝説

ご好評につきお申し込みは、お早目にお願いします。

ニューパッケージで新登場!!

3-Dゴルフシミュレーション FM-7/NEW7/77 Part2

テープ版 ¥4,000

- ★完全機械語により、画面表示が高速化。
- ★ホールデータ 一括ローディングにより、スピーディなゲーム展開。
- ★2人でのプレイ可能。
- ★ボール拡大機能（ボールの位置が見やすくなりました。）
- ★プラクティス機能（ワンホール毎に繰り返し練習できます。）
- ★ステップ機能（ショットしたボールを、コマ送りできます。）
- ★オートリピート機能（全ホールの完全自動再現可能。）
- ★ゲーム中断機能（ラウンド途中からの再プレイ可能。）

T&E SOFT

★T&EマガジンNo.3ご希望の方は、100円切手2枚（200円分）同封の上、マガジン請求券をお送り下さい。
★T&E SOFTの新しいカタログ（No.2）ができました。ご希望の方は100円切手同封の上、カタログ請求券をお送り下さい。
★通信販売ご希望の方は現金書留で料金と商品名・機械名を明記の上当社宛お送りください（送料サービス・速達希望の方は300円プラス）

7月30日に 事務所が移転しました。電話番号は従来通りです。

新住所 〒465 名古屋市名東区豊が丘1810番地
TEL 052-773-7770（代）

T&Eマガジン No.3 請求券 Oh!FM 9月号

カタログ No.2 請求券 Oh!FM 9月号

新発売

ジェルダ戦役 第2弾！

JELDA II

3Dリアルタイムアクション＋SFアドベンチャーゲーム

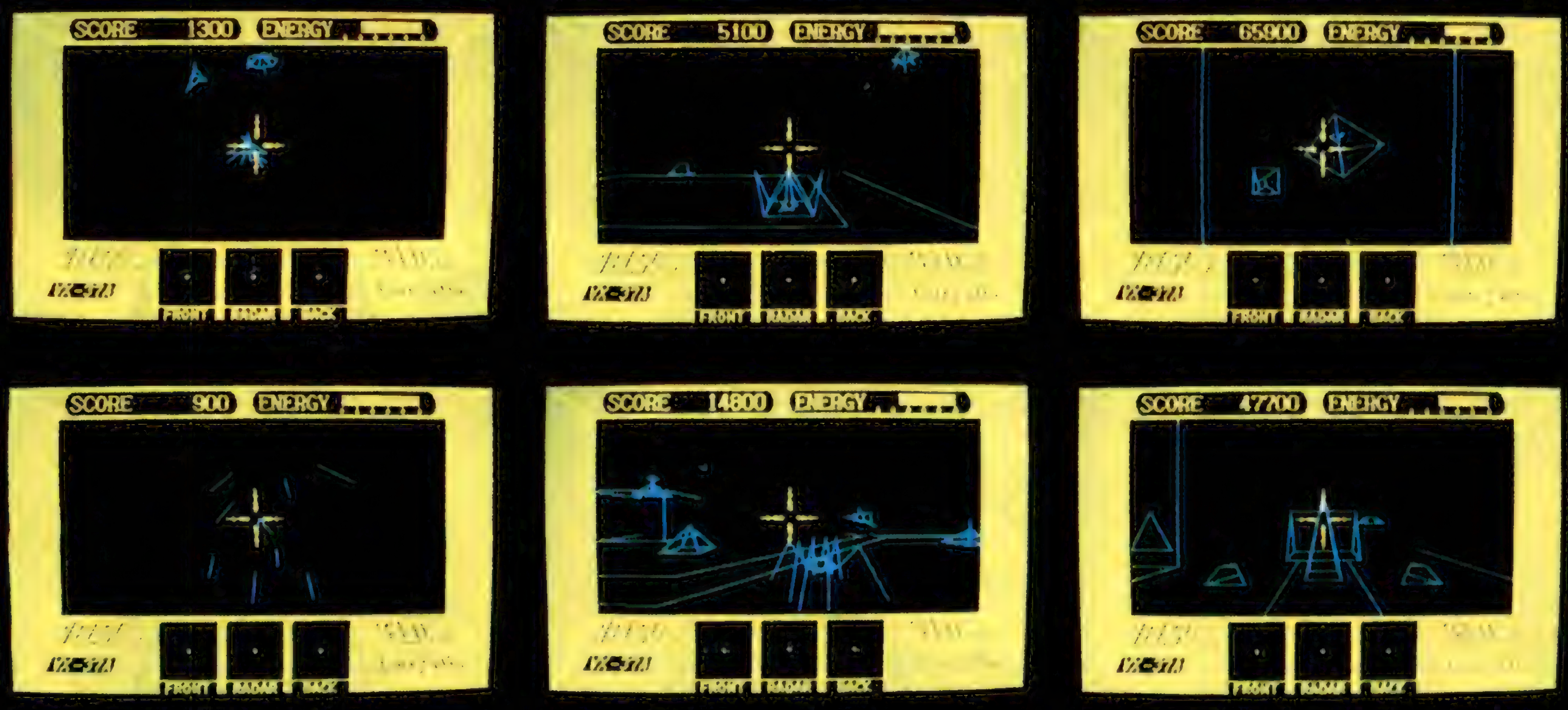
作戦暗号名 V（オペレーションコード ファイ）

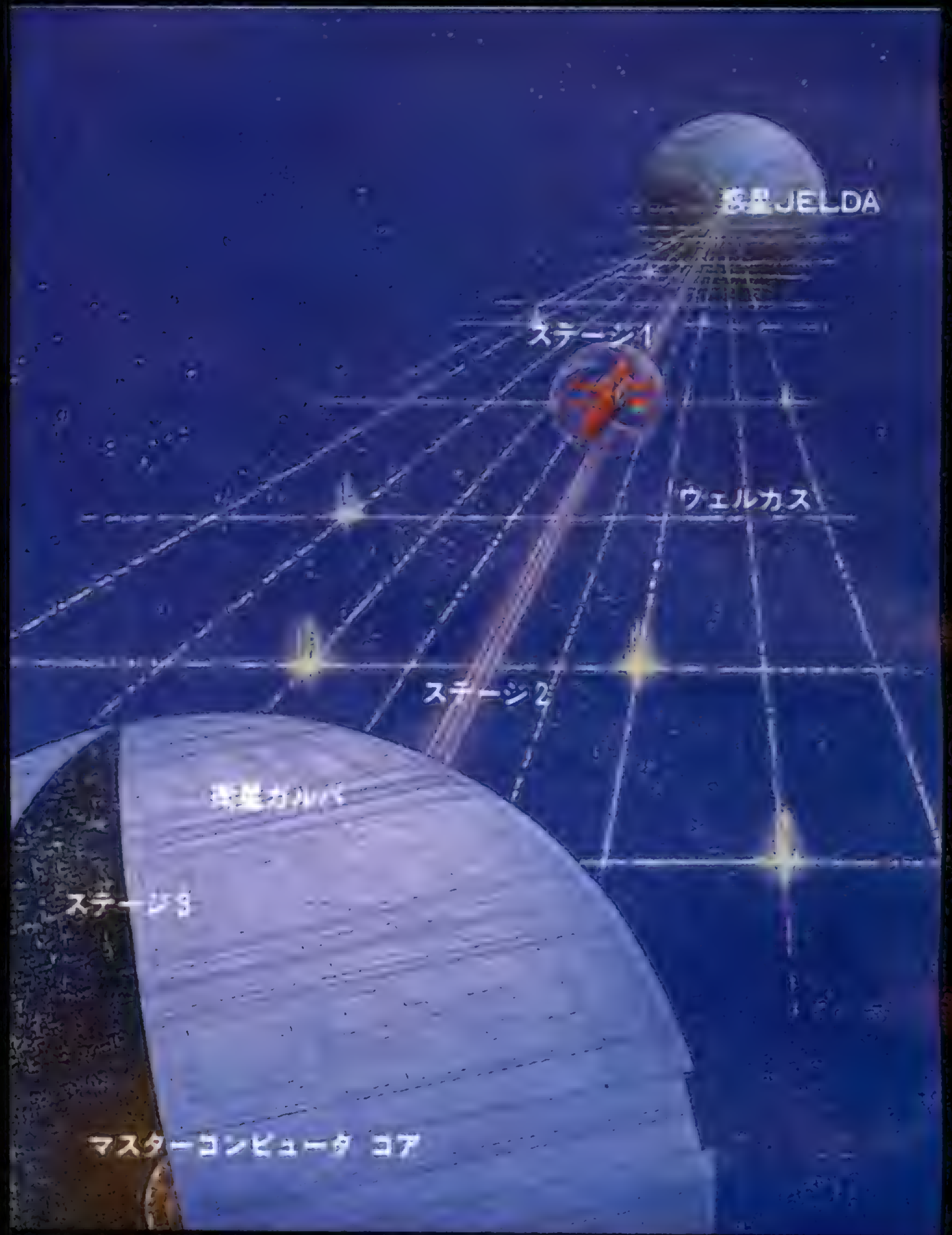
エネルギーシールド装置を強化したヴェルガスを使用し、直径1kmにわたる巨大秘密衛星ガルバを破壊し、敵国に決定的なダメージを与える事が目的である。

STAGE1 惑星ジェルダから発進し、ガルバ前衛部隊をせん滅する

STAGE2 衛星ガルバ上で戦闘し、内部進入路ゲートを発見する

STAGE3 内部迷路を進み、メインコンピュータ「コア」を破壊する





(CT)¥3,800(FD)¥5,800

FMシリーズ/PC-9800シリーズ/PC-8800シリーズ/X-1シリーズ/MZ-2200シリーズ

日本語ワードプロセッサー FM-7/8 FM-NEW7

JET FM

FMユーザーの夢を実現！

1. 操作性 使いやすさと実用性を重視！

2. 高機能 他と較べて下さい。

3. 低価格 26,800円

■JET仕様一覧

5FD.2枚 定価 26,800円

●**文書エリア**（40字×96行）●**テキスト表示**（40文字×10行）●**文字種類**（漢字 JIS第一水準2965文字＋107文字 記号 768文字 外字 96文字）●**文字サイズ**（全角文字、倍角文字、半角文字、1/4角文字 （123＋−789）●**辞書**（約3万語のかな＝漢字変換方式（熟語単位）使用頻度順学習機能付、約4万語まで拡張可能）●**編集**（文字・行の挿入・削除、桁の挿入・削除、行そろえ、40文字までの文字の切り貼り、ブロック複写、センターリング、右寄せ、左寄せ、タブ、インデント機能、文書復活機能 etc.●**文書記録**（1ディスケットに最大64文書記録、パスワードによるによる機密保護可能●**文書印刷**（縦書き・横書き印刷、行間隔・文字間隔設定、印刷枚数、連続ページ印刷）●**定形句管理**（40文字×96句、定形句の定義、内容一覧など）■**ユーティリティー**（**外字管理**）（外字の作成、外字の内容一覧）（**MZ文書呼出し**（JET-MZで作成した文書を使用できます。）**文書通信機能**（電話回線を通じての文書転送（オプション）

Carry lab. キャンペーンプレゼント当選者発表

〈A賞〉
神奈川県川崎市 石沢 伸康

〈B賞〉
東京都足立区 関 邦一
栃木県小山市 髙橋健二

〈C賞〉

愛名県豊橋市	岩尾光輝
新潟県中頸城郡	植木正男
茨城県新治郡	内野哲男
大阪府吹田市	加藤圭一
東京都東村山市	川口広海
大阪府八尾市	川崎哲也
愛知県名古屋市	北川浩史
岡山県倉敷市	高原邦緒
大阪市東城区	福田克正
滋賀県野洲郡	中原武則

チャレンジャーズ ソフト情報誌「チャレンジャーズ」ご希望の方は、氏名、住所、年令、職業を明記の上、送料200と請求券を添えてお申し込み下さい。

通信販売 全国マイコン販売店で取り扱っております。通信販売の御注文は現金書留か郵便振替を御利用下さい。郵便振替口座/熊本0-18846

サポート 月〜金曜日14:00〜17:00 TEL 096-363-0047

Carry soft. © Carry lab.

㈱キャリーラボ 〒862 熊本市大江6丁目25-25 金子ビル1F
TEL 096-363-0211 FAX 096-363-0235-G2・G3

エニックスソフトウェア大賞

第3回ゲームホビープログラムコンテスト

新たな"アーティスト(印税所得者)"の幕開けだ。
その名は『ゲームプログラム作者』
小説家や漫画家、それに作詞家・作曲家のような――
夢のある"印税生活"が君を待っている。
パソコンひとつで富と名声をつかめる時代がやってきた。
中村光一君、19歳。
今年3月の1ヵ月間で4,067,220円もの印税を、エニックスから受けとった彼――
「ドアドア」一本だけで、印税額は2,000万円を越えるだろう。
アメリカで、いや、日本で続々と誕生する"ゲーム作者のリッチマン"達。
君にも、そのチャンスがやってきた。
さあ、今こそチャレンジの時だ!

エニックスソフトウェア大賞 500万円

- ●エニックスソフトウェア大賞……トロフィ・表彰状・金500万円(1名)
- ●優　秀　プログラム賞……………楯・表彰状・金250万円(2名)
- ●入　賞　プログラム賞……………楯・表彰状・金100万円(10名)
- ●参　加　賞……………………………記念品(応募された方全員)

※上記の金額は、印税の最低保証金であり、入賞者に全額前渡しされます。

●募集期限
昭和59年9月15日(当日消印有効)

●募集内容
a)ゲームホビープログラム
b)作品は、新聞雑誌等に未発表で販売流通していないオリジナルに限ります。
c)使用機種は、市販されているパソコン・マイコンであれば特に限定しません。

●応募方法
a)個人、グループ、法人等いずれでもけっこうです。
b)所定の申し込み書と、プログラムを収録したテープまたはディスケットを添えてお送り下さい。
尚、申し込み書は主なパソコンショップ店頭か、下記へハガキもしくは電話でご請求下さい。

〒160 東京都新宿区西新宿7-1-8
☎03(371)5581
㈱エニックスプログラムコンテスト事務局

c)応募作品の著作権は作者に帰属します。
d)商品化に際して、入賞作品の独占販売権は当社に帰属します。
e)印税は、当社規定によりお支払いいたします。

●発表
昭和59年10月31日以降に発売される主なマイコン雑誌誌上。
※応募者には直接ご通知いたします。

主催 株式会社エニックス

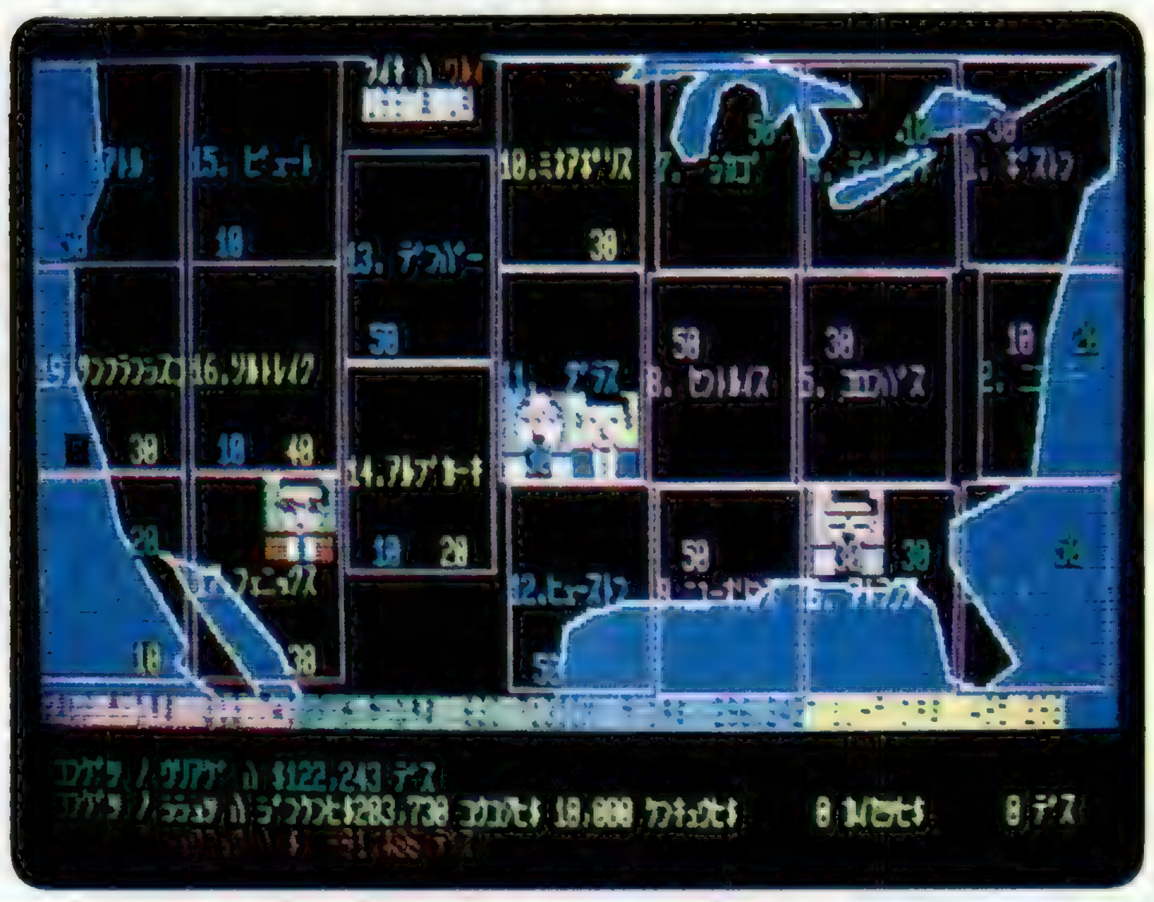
●FM-7 DSK PWUP UT（ユーティリティ）

●High-SPEED SAVE（3～5倍）
●日付記録機能
●RAM DISK
●裏RAMをプリンタバッファとして使用
●新コマンドの追加

●STAIR CAT（ゲーム）

がけから落ちた子猫が、落下物を避けながら地上まではい上がる。体力メータに気をつけ、パワーエサを食べながら前進。一面クリアで一匹加算。

●BUSINESS TOUR（ゲーム）



●ビジネスツアー

アメリカ市場を舞台に、あなたの経営センスで、コンピュータメーカーの社長として、会社を大きくすることができるかどうか、楽しみながら、経営戦略をシミュレートするいままでにないビジネス・思考型ロールプレイングゲーム。

FM-7/8/NEW-7	(CT)	￥4,200
PC-8801/mkII	(CT)	￥4,200
	(5FD)	￥6,200
X1/C	(CT)	￥4,200

★お楽しみプレゼント★ 当社製品の応募シールをお送り下さい。毎月抽選で豪華な当社オリジナルグッズをプレゼント致します。

※通信販売も行なっております。ご注文の際は、品名・機種名・住所・電話番号を明記の上、必ず現金書留でお申込下さい。（送料無料）尚、お急ぎの方は、速達代金300円追加して下さい。

企画・開発スタッフ募集中!!
（詳細につきましては、電話でお問い合せ下さい。）

コンピュータのよりよい活用をめざす。
BOTHTEC inc.

ボーステック株式会社
〒116 東京都荒川区西日暮里2-41-16
TEL. 03(801)2301(代表)

X1・FM-7シリーズ ユーザーに告ぐ―

これが全米最強ソフトだ！

from U.S.A.
Licensed from
Brøderbund

かつてない面白さが、遂に日本上陸！

Lode Runner

新発売

X-1・FM-7 シリーズ
ロードランナー

カセットテープ版 ￥4,800　フロッピィ版 ￥5,800

150もの地下迷宮と、プラス無限のバリエーション。その中で、隠された宝を奪い返さねばならない。敵は迫る。ロードランナーは、地上への出口を求めてひたすら疾走する――― 発売以来、全米が熱狂。いまもなお、ヒットチャートを走り続ける「ロードランナー」。凄い！面白い！これはまさに、無敵のソフトだ！

CHOPLIFTER!

「64人の人質を奪還せよ」それが君への使命だ。次々と君を襲う敵の攻撃網をかわして、見事ヒーローになれるか。

X-1・FM-7 シリーズ　カセットテープ版 ￥4,800　フロッピィ版 ￥5,800

新発売

チョプリフター

Spare Change

さあ、大変だ！2人のザークが、お金をかき集めて脱出しようとしている。ハイテクニックで、彼らを逃すな！

X-1・FM-7 シリーズ　カセットテープ版 ￥4,800　フロッピィ版 ￥5,800

近日発売

スペアチェンジ

Soft Pro International

〒530 大阪市北区西天満6-7-2 梅新東ビル5F TEL(06)363-1221

●お買い求めは、全国パソコンショップで！
販売 株式会社小西六エニックス
制作 ソフトプロ株式会社
ソフトプロ インターナショナル事業部

ジャイアンツのかたきは、僕がうつ。

新発売

適応機種　X-1シリーズ　PC-6001MKII　PC-8801シリーズ　FM-7　TAPE版 ¥4,000　FD版 ¥6,800

部屋は、ホームグラウンド

野球狂

くやしいだろう、逆転さよなら負けなんて。
だから、僕がさっそくかたき討ち。
まるで、テレビの野球中継みたいに、選手も
審判も動く「野球狂」。プロ野球や高校野球、草
野球など、自分の好きなチームと対戦チームの選手それ
ぞれの打率や守備率などのデータを入力。それに従って動く画面で、
リアルタイムゲームが楽しめるのだ。近頃は、僕が寝た後、父さんが
自分の草野球チームとジャイアンツ、試合させてるみたいだよ。

HUDSON GROUP
HUDSON SOFT®

本社・ハドソン札幌/〒062札幌市豊平区平岸3条5丁目4番17号 コロナード平岸II 201 PHONE:011-821-1538
営業所・仙台・金沢・東京・大阪・福岡・沖縄

Nintendo®
©1983 Nintendo of America Inc.
新発売
寝不足にして、ワルイ、ワルイ。
面白すぎて、やめられない。
MARIO BROS.™
SPECIAL
好評発売中
宇宙やジャングルを舞台に、ハチのバズビーやおじゃま虫クリーピーたちを引き連れたドンキーコングが大暴れ！さあ、お得意のスプレーで、負けるな、スタンリー。
HI SCORE 500
SCORE 30
STAGE 1
STANLEY
HUDSON SOFT
Nintendo
1984
WRITTEN BY ITA
敵は、可愛いカメやカニ。さて、マリオは、彼らを蹴飛ばして、最上階までジャンプできるかな。
面白すぎて、とまらない。
DONKEY KONG 3
大逆襲
HUDSON GROUP
HUDSON SOFT®
適応機種　X-1シリーズ　PC-8801シリーズ　FM-7シリーズ　PC-6001MKII　PC-6601　TAPE版 ¥3,600　FD版 ¥5,800
本社・ハドソン札幌／〒062 札幌市豊平区平岸3条5丁目4番17号 コロナード平岸II 201　PHONE：011-821-1538
営業所・仙台・金沢・東京・大阪・福岡・沖縄

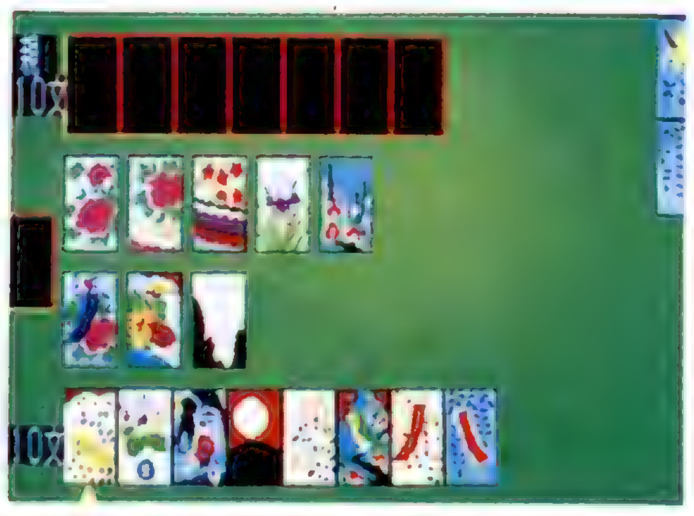
実戦ギャンブラー。

切り札は、これだ。

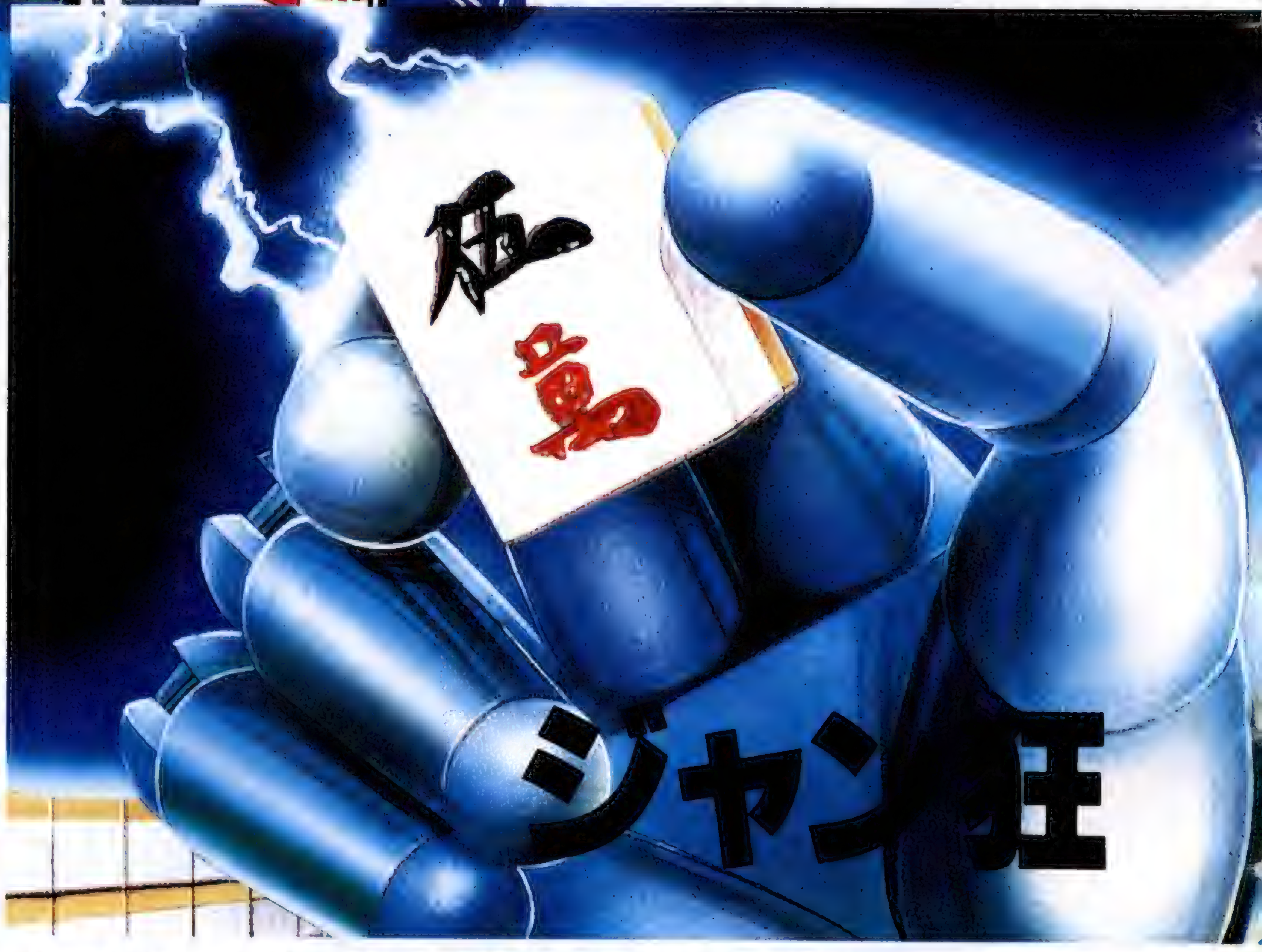
コンピュータと、ブラックジャックをお手合わせ。明日のデートを、カードで占う。『トランプ狂』は、ブリッジ、神経衰弱、ポーカーなど数々の遊びができるワクワクソフトだ。 X-1・FM-7・PC-88シリーズ

コンピュータも、花見で一杯。

こいこい、ばかっ花、おいちょかぶ。花札通はもちろん、初心者もオープンモードにすれば、大丈夫。さあ、人間のように考えるコンピュータと、勝負!勝負!

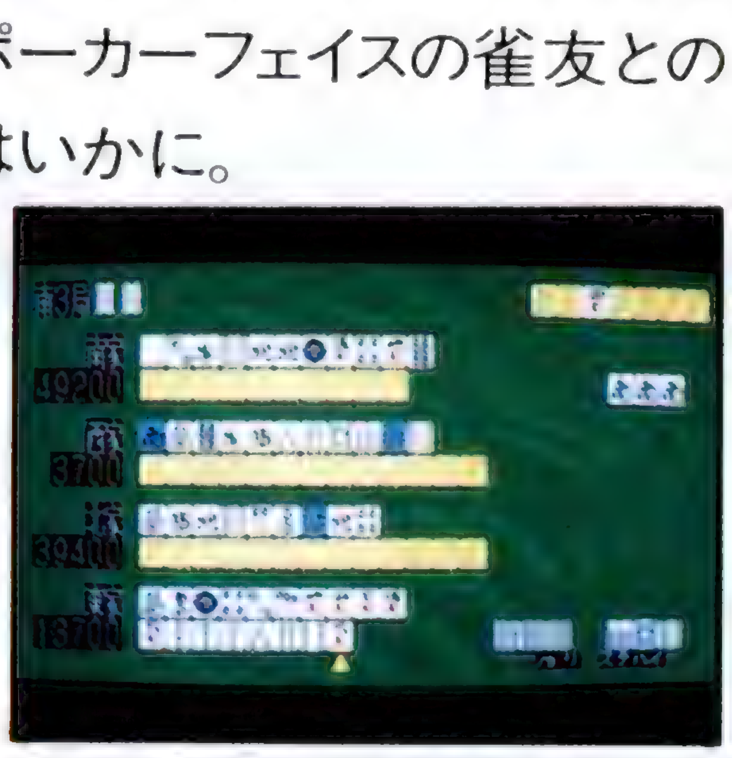


X-1, PC-8801, 9801, FM-7
●テープ版￥4,000, ディスク版￥6,800



役満をねらえ。

まるで実戦さながらの対局感! コンピュータは、ほぼ人間と同じ思考プログラムだから、清一色や混一色への決断はもちろん、国士無双などの役満もねらってくる。さて、ポーカーフェイスの雀友との勝負はいかに。



適応機種:X-1, PC-6001, 8801, PC-9801F, PC-9801, FM-7, FM-11, S1, MZ-5500, 1500
●テープ版￥4,000, ディスク版￥6,800, QD版￥4,800

HUDSON GROUP
HUDSON SOFT

本社・ハドソン札幌/〒062札幌市豊平区平岸3条5丁目4番17号 コロナード平岸II 201 PHONE:011-821-1538 営業所・仙台・金沢・東京・大阪・福岡・沖縄

サラダの国のトマト姫

150枚を越えるときめきのファンタジック・アドベンチャー

ハラハラドキドキ ストーリー

重い年貢をとりたてて国民を苦しめるカボチャ大王。彼に捕えられた反乱軍のリーダー、トマト姫。さて、絶体絶命のピンチの最中にさっそうと現れたキュウリ戦士は、トマト姫を無事救出できるか。反乱軍を勝利に導けるか。

適応機種

X-1シリーズ	3″FD
PC-60MKII、66	T
FM-7	T/5″FD
FM-77	3.5″FD
PC-8801/MKII	T
	5″FD
PC-9801 F	5″FD

T ¥4,800 FD ¥6,800

胸踊る数々の冒険、そしてキュートな野菜の主人公が織りなすさすらいのロマン。人気絶頂の竹・中コンビが、壮大なアドベンチャー・ストーリーを自信をもってキミに送る。

思わず興奮アドベンチャー

好評発売中

5つのパビリオンのどこかに、あの秘宝「三月磨臼(ミツキマウス)」が隠されている。ヒントは、114枚の画面すべてに隠されているキーワード。

さて、キミは この謎解きに何日かかるかな。

推理小説にも負けない面白アドベンチャー

デゼニランド

適応機種

PC-6001/MKII	T	XI-C	T
PC-8801/MKII	5″FD	XI-D	3″FD
PC-9801/F	5″FD	FM-7	5″FD、T

T ¥4,800 5″FD ¥6,800 3″FD ¥5,800

各ゲーム、ついにS1搭載！

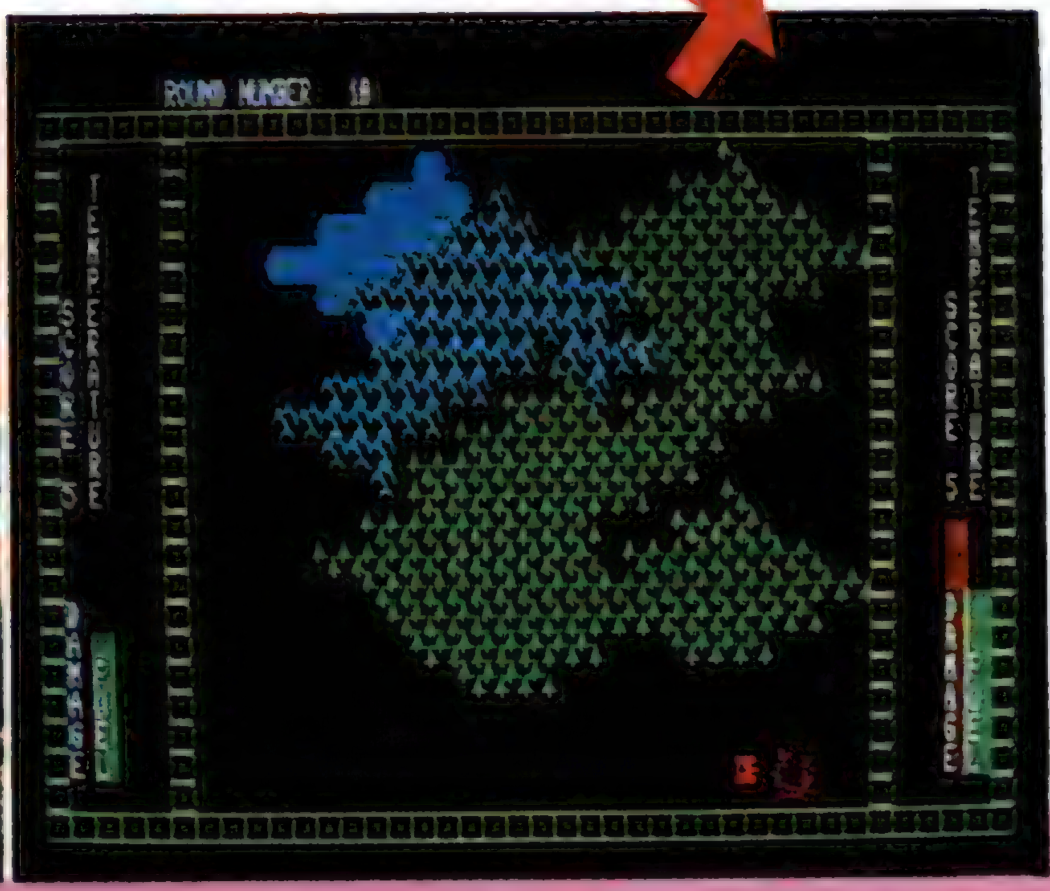
HUDSON GROUP
HUDSON SOFT
本社・ハドソン札幌 〒062 札幌市豊平区平岸3条5丁目4番17号 コロナード平岸II201 PHONE：011-821-1538 営業所・仙台・金沢・東京・大阪・福岡・沖縄

アメリカン・エスプリを満載
した面白ゲーム第二弾新発売

アメリカ生まれ
SilverBall

アドベンチャーランドあり、
ファンタジィランドあり、
グラフィックスもエンジョイできる
SilverBall-landに強力ゲー

敵船団発見！機雷・爆雷・大イカを
シー・
SEA



森や川や荒野での戦車対戦車の凄絶な戦い、君は生き残れるか？
タンク・ウォー
TANK WAR
商品コード番号　FUSK23010

朝の静寂を破っての敵襲だ！
恋人のいる街を防衛せよ。
ドーン・パトロール
DAWN PATROL
商品コード番号　FUSK23002

魔王の黄金を奪え！オオムに乗って
ヒーローは飛び立つ。
クラウド・キングダム"雲の王国"
CLOUD KINGDOM
商品コード番号　FUSK23007

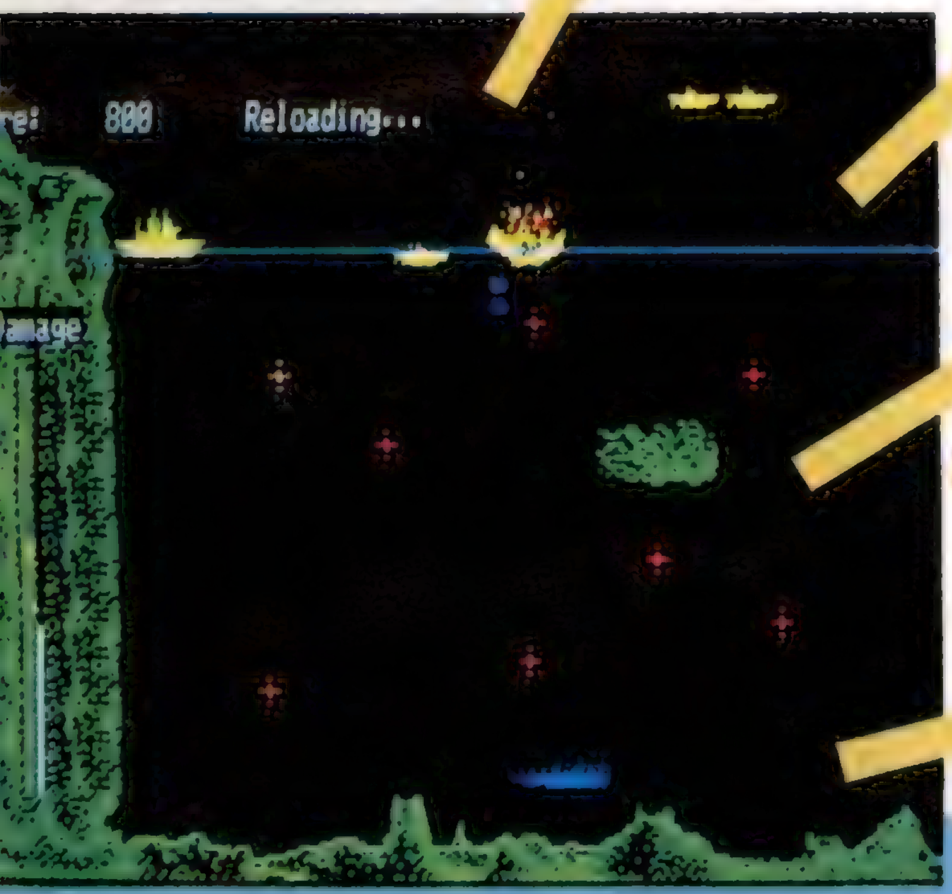
外敵から、飢えからヒナを守れ！
ハヤブサ愛情物語。
ファルコン・アドベンチャー
FALCON ADVENTURE
商品コード番号　FUSK23008

FM-7用 COLOR/USA版

スピード・スリル・サスペンス、これに「あそび」の
アイデアが加わったアメリカ生まれの面白ゲーム
〝SilverBall〟。FM-7の世界に初登場。
好評発売中です。

アメリカ生まれ SilverBallはスペクタクル空間

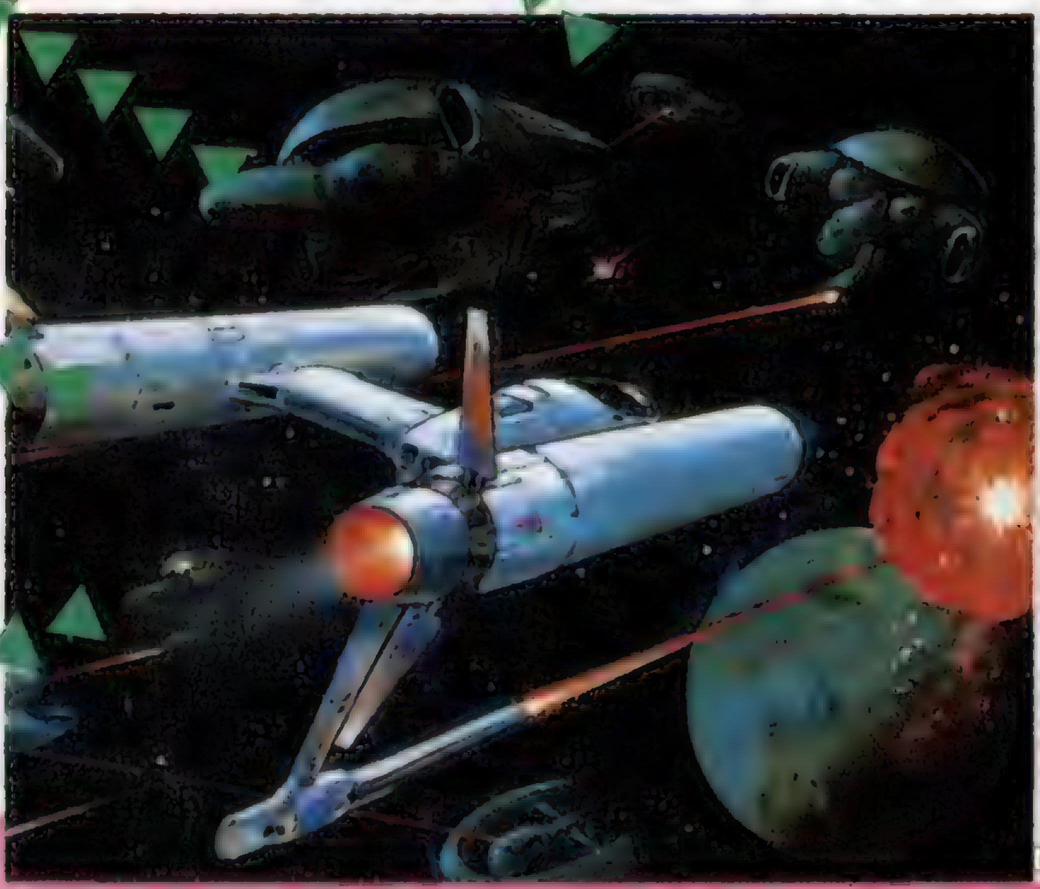
ム出現。



さけて急浮上、魚雷攻撃で全滅せよ。

ハント
HUNT

商品コード番号 FUSK 23009



しつこく迫るUFO軍を撃退し、惑星めがけ針路をとれ！

デス・プラネット・バトル
DEATH PLANET BATTLE

商品コード番号 FUSK 23001

敵の砲弾飛びかうなか、
輸送軍団をやっつけろ！

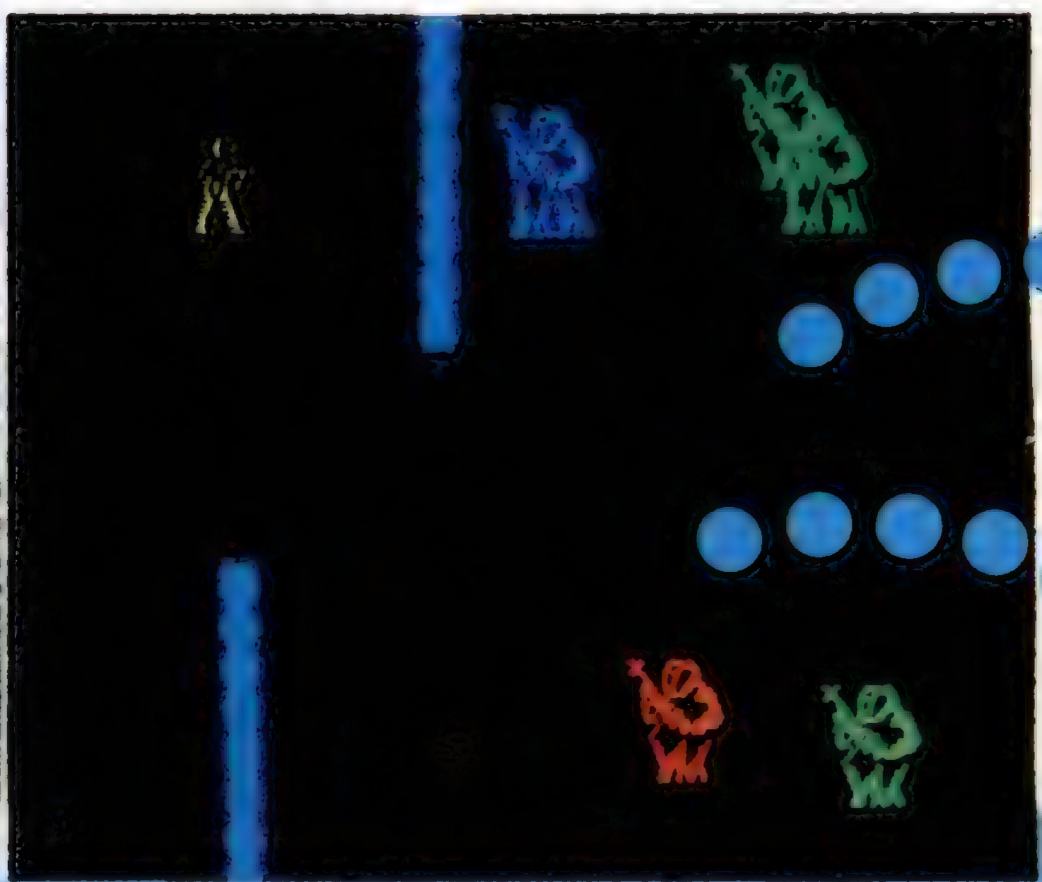
フライボーイ
FLYBOY

商品コード番号 FUSK23005

攻撃!!敵戦車に占領された街を奪回せよ！

レーザー・タンク
LASER TANKS

商品コード番号 FUSK23011



殺るか殺られるか、スリリングな銃撃戦に勝ち残れ！

ロボット・ウォー
ROBOT WAR

商品コード番号 FUSK23006

FM-7用カセットテープ版
定価 各3500円

製造・発売元 シルバーボウルソフトウェア
SilverBall
2150 Shattuck Ave. Suite 902 Berkeley, CA. 94704 U.S.A.
お問い合せは…… 株式会社日本ソフトバンク
〒102 東京都千代田区九段南2-3-14 靖国九段南ビル
電話 03(263)3904

通信販売しておりませんのでパソコンショップでお求めください。
＊このテープよりの複製及びレンタル営業行為を禁止します。

SilverBallは次々と新しいゲームを
発表・発売していきます。
ご期待ください。

『ちょっと大きい手を狙うと、安あがりするのが憎いね。』
『ヤミテンやヒッカケ・リーチだってやるしね。』

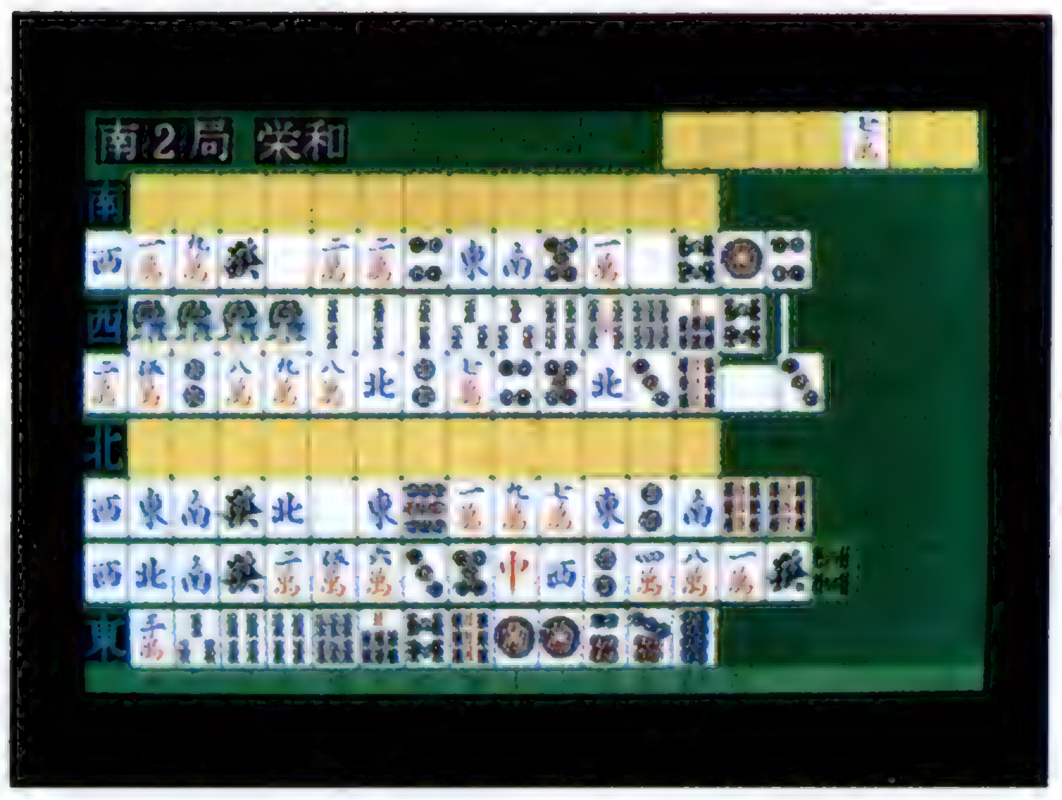
最強レベルでコンピュータと戦ったベテラン雀士の感想です。

〈プロフェッショナル麻雀〉は、アセンブリ言語で記述され最高速で動く本格的なゲーム・ソフトです。麻雀の忠実なシミュレーションに基づいて組立てられており、ゲームは実戦そのままに展開します。

鮮明に、美しく表示されるリアルな牌。あなたの入力のスピードのままに、快適なテンポでゲームは進行します。3人の対戦相手の実力レベルは、3段階に変えることができ、高位レベルでは最強の打ち手として登場します。和了すると点数は自動計算されて画面に表示され、半荘が終了するとそれぞれの点数の合計が瞬時に行なわれます。スイッチを切るまで得点は累計されて行きますから、勝つまで徹夜マージャンということもありえますね。

- リーチはもちろん、つみ場、裏ドラ、五本場以上二翻しばりなど、ルールは実戦そのままです。
- 実際にある麻雀の役は、すべて網羅されています。
- 新しい思考ルーチンの開発によって、コンピュータの実力は最強に設定できます。最高レベルでコンピュータに勝てれば、あなたの実力はかなりのものです。
- 対戦相手のスピードは8段階に設定でき、食い断、不聴罰の有無も選択できます。
- チー、ポン、カン、ロンができるときには、チャイムが鳴ってゲームの進行が止まります。実戦の緊張感を味わいたいときには、停止しない実戦モードに切り換えられます。
- コンピュータの手牌を見ることができるオープン・モード、コンピュータだけで対戦するデモ・モードもあります。
- 同じ配牌からの再ゲームも可能で、初心者の実力アップに役立ちます。

●マイコンショップで好評発売中!!



メディア／機種	8" ¥7,200	5"2HD ¥7,200	5"2DD ¥6,800	5"2D ¥6,800	3.5" ¥6,800	テープ ¥4,800
PC9801/E	●			●		
PC9801F			●			
PC8801(mKII)	●			●		
N5200 モデル05	*●					
FM-11AD2/BS		●				
FM-11AD/EX				●		
FM77					●	
FM7, FM New7				●		●
X-1						●
SMC70・SMC777					●	

＊N5200用の8"のみ¥9,800.です。

プロフェッショナル 麻雀

●48Pマニュアル付

Chat Noir®

a guarantee for software that won't wear

株式会社 シャノアール

〒102 東京都千代田区隼町2-12 藤和半蔵門コープ405号
Tel. (03)234-0950

16ビットパソコンのハードおよびソフトの総合活用をわかりやすく紹介する実用情報誌。

創刊1号の主な内容

特集 マウスの機能に接近!!
APソフトの現状分析からマウスの未来まで。

dBASE II による人工知能"D-Brain II"の開発

ハイパワー連載
- MACHINEテストランレポート
- dBASE II 入門
- SOFT WATCHING
- 統合ソフトの活用事情'84
- パソコンテレコミュニケーション総カタログ

日本ソフトバンク出版部発行　☎03-261-4095

創刊記念大募集

「BEEP」は、みんながつくるパソコンゲーム情報誌です。待望の創刊を記念して、下記のとおり「プログラム」「原稿」を募集します。ふるって投稿してください。

●あて先
〒102東京都千代田区四番町2-1
日本ソフトバンク出版部「BEEP」編集室

〈ゲームプログラム〉

ゲームのジャンルは問いません。未発表のオリジナルゲームをどしどし投稿してください(ショート、ロングいずれも可)。

●プログラムはテープに入れてください(投稿作品は原則としてお返しできませんので、あらかじめコピーをとってください)。
●プログラムの内容の説明、操作方法、遊び方を、400字づめ原稿用紙3枚ぐらいにまとめてください(原稿には、対象機種、使用言語、住所、氏名、年齢、学校名、連絡先電話番号を明記してください)。
●創刊号(1月号)掲載分の締切は、10月末日です。
●掲載されたプログラムには本誌規定の原稿料を支払い、掲載誌を送ります。

〈原稿〉

「VOICE OF BEEPER」と「はみだしBEEP」の二つについて、原稿を募集します。

①「VOICE OF BEEPER」は、ゲームについて、みんなの声と顔写真を載せるページです。たとえば、「これは楽しい!ゼッタイおすすめソフトだ」、「がっかり!期待はずれのソフト」、「ゲーム言いたい放題」、「ここが知りたい、だれか教えて」などなど、ゲームについてならなんでもO.K.。みんなの声を書いてください。
●原稿は400字づめ原稿用紙1枚ぐらいをめやすにしてください(多くても少くても可)。
●顔写真(なるべくモノクロ。サイズはサービス判くらい)を添付してください。

②「はみだしBEEP」にも、みんなの声を寄せてください。
●原稿はなるべく100字ぐらいにしてください(多くても少なくても可)。

★①については、原稿採用の場合は、「BEEP」特製トレーナーをプレゼントします。
★①、②とも創刊号の原稿締切は、9月末日です。
★①、②とも、原稿はたて書きにしてください。

SOFTBANK 日本ソフトバンク出版部

あなたOS知ってますか

田奈　真実

うーアツイ。あついなあ。なんとかならんのかなー。僕，実に汗っかきで，すぐあせもができるんですね。あー泳ぎにでも行きたいなー。僕の下宿，クーラーも扇風機もうちわすらないんです。だからカナワンのですよ，暑さが。この原稿を書いたら扇風機くらい買えるかねー，Mさん。この暑い夏をのりきるには，まあ，クーラーとはいわないまでも，せめて扇風機くらい欲しいところ。風とおしが悪く，西日の強い4畳半の貧乏下宿を，少しでもよい環境にするツールが欲しい。クソ暑いへやが涼しくなれば，プログラミングが効率的に進むんだけどなー。

さて，話はコロッと変わりますが，クーラーや扇風機ではないけれど，マイコンユーザに快適な使用環境を提供してくれるものにOS（Operating System）というものがあります。OSによって，プログラミング環境はバツグンによくなるのでありますよ。あなた，知ってますか。

はははっ，いきなり強引な導入をしてしまいました。今月は，この「OS」を特集しました。最後まで，ごゆるりとお楽しみください。

OSってナンデスカ

OSとは，簡単にいってしまうと，周辺装置（ディスク，プリンタなど）を効率的に利用できるように全体を管理するソフトウェアであり，コンピュータを使いやすくする道具です。OSにより，なぜ使いやすくなるのか，これから説明いたします。ハイ。

OSとDOSはどう違う?

さて，話を進める前に，OSとDOSの違いに触れておきましょう。

OSはOperating System，DOSはDisk Operating System*1です。DOSは，対象をディスクに限っているといってもよいでしょう。OSの場合，オペレーション（操作）の対象は，システムを構成するすべての装置で，システム全体の効率的運用が目的です。DOSの場合，ディスクとオペレータの間のみを重点的にみています。このようにOSとDOSは，システムの規模とか，管理する対象・目的が違い，DOSはOSにふくまれてしまうものといってよいでしょう。パソコンでは，OSとはほとんどの場合，DOSをさしています。

OSを使うとなぜ便利か

プログラムを組み合わせて使用できる

OSでは,「エディタ」,「コンパイラ」といった,いろいろなソフトウェアを組み合わせて使えます。これがなぜ便利か説明しましょう。

プログラムを実行するときのことを考えてみてください。実行には,次の2つのステップが必要です。

①プログラムを打ち込む。

②プログラムを実行する。

このように2段階で終わるのは,実行がインタプリタによって行われる場合です。コンパイラを使う場合は,②以降が,

②コンパイルする*2。

③リンクする*3。

④実行(ロードを含む)する。

となります。このように,プログラムを走らせるというのは,いくつかのステップに分けることができます。パソコンでは,もっともポピュラなBASICという言語の場合,このようなステップ分けが厳密にできません。BASICは,インタプリタ自身が①のステップを含んでいるのです。本来は,①のステップはプログラムを打ち込むだけ,つまりパソコン(のディスク)に文字を書き込むだけの作業で,これを行うプログラムを「エディタ」なんぞと申します。BASICでは,エディタがプログラムを実行するプログラム(書かれたプログラムを実行するプログラム,すなわちインタプリタ)といっしょになっているわけです。

さて,いろいろな言語の持つ特長を生かして,それぞれの言語で書かれたプログラムを組み合わせて使いたいとします。各言語ごとにエディタが存在したら,言語ごとにプログラムの打ち込み方が違ってしまいます。実際には,何種類ものエディタの使い方を習得するのは至難の技でして,うそだと思うなら,Mince, WS, ex, ED, vi, EMACSといったエディタをすべて使ってみるとよいでしょう。

このような不都合を避けるために,各ステップごとに,プログラムが分かれていたほうが便利なのです。たとえば,別の言語を使いたいときは,②のプログラム(コンパイラ)だけを取り換えればよいわけです。

このように各ステップごとに独立したプログラムが存在するとすれば,「プログラムを組み合わせて使える」ことが必要なわけです。

また,この機能は同一の言語に対しても有効です。Cという言語があります。この言語ほど多くのメーカからコンパイラが発売されているものはありません。いままでA社のCコンパイラを使っていたのだがB社から新しく出たCコンパイラの方が実行が速い,といった場合,コンパイラを取り換えてしまえばOKです。ここでも上記の機能が威力を発揮します。

言語によらず基本操作が共通化される

DOSを使用した場合,言語の種類にかかわらずディスクの基本操作は同じです。基本操作とは,ロード,ディスクに入っているファイル名を見る(BASICではFILES),ファイルの内容を表示,印刷する,空き領域の量を見るなどです。さらに進化したDOSでは,ディレクトリのサポート,マルチタスク,マルチユーザ関係のサポートなども基本操作に入ります。DOSを使うことにより,これらの操作が共通のものとなります。

機種が違っても操作は同じ

パソコンが違っても,そこで動かすDOSが共通ならば,操作も共通です。FM, PC, MZを問わず,使用するDOSが同じなら,ユーザは同様に扱えるわけです。

しかし残念ながら,これは操作・使用法に関してのみいえることなのです。たとえばCP/M-86という DOS。今までFMで使っていたCP/M-86の5インチディスクをPCのCP/M-86で読もうとしても,まったく読めません。これは機種によりディスクのフォーマット(正しくはディスク上のロケーション)が違うことに起因する問題です。同じDOSでも,それが動いている機種によってディスクの構造が違ってしまい,ディスクに互換性がないものが多いようです。しかし,CP/MのようにMZとFM, PCのディスクが共通に使えるものもあります。

さて操作が共通ということは,別のメリットを生じさせます。機種によらず同じDOSが動くのですから,DOSのもとで作成されたプログラムは機種を選ばずに走ることになります。つまり,FM用,PC用といった機種別のプログラムを作らずにすむのです。ただディスクのフォーマットが,多くの場合違っていますから,それぞれのフォーマット用にセーブしておかなければなりません。しかしプログラムを作り直す必要はないわけです。これにより,ソフトウェアの低価格化も可能となります。

といっても,プログラムは機種別のハードウェアに依存してはならないという制約があります。たとえばグラフィックやサウンドなどの機能はDOSの制御下にない*4ので,当然のことながらプログラムが何もかも自分で処理しなければなりません。ということは,そのプログラムは,そのハードウェア(パソコン)でしか動作しないということになります。

DOS独自のサービス機能

DOS自体の機能が向上してきますと,ユーザに対するサービスが増えます。たとえば**マルチタスク**。このような機能を持ってくるとDOSというよりもOSに近いものになります。これは複数の仕事を見かけ上同時に実行できる機能でして,このような

機能はDOS(OS)独特の機能になり，個々のプログラムではサポートできません。複数のプログラムに対してのみ有効な機能だからです。たとえばプリンタにリストを打ち出す一方で，プログラムを実行する。アセンブルする一方で，プログラムの修正を行う。一方でファイルのソートをしながらプログラムの入力をする。こんなことが可能となると，ユーザは実によい使い心地を得られます。

ソフトウエアバスとしての機能

かつてDOSはソフトウェアバスとしてもてはやされていました。これはOS上のソフトウェアが機種を問わず走ることを意味しているわけです。しかし前述のとおり，ディスクのフォーマットの相違などにより，必ずしもソフトウェアバスとはいえなくなっており，最近ではこのようには呼ばれなくなってきています。ただMS-DOSでは，ディスクのフォーマットが共通になっている場合が多く，この機能を保っています。

OSの歴史

さて，OSの概念がもう少し整理できるよう，OSの歴史をふり返ってみましょう。

現在のコンピュータ，これはノイマン型コンピュータ*5と呼ばれます。この型のコンピュータは，1949年ケンブリッジ大学で作られたEDSACが最初になります。日本では1956年通産省電気試験所で，ETLマークIIIが最初に作られました。この頃のコンピュータには補助記憶装置はなく，OSが管理する対象もありませんでした。1950年UNIVAC I*6で初めて磁気テープ°*7が使用されたのです。記憶容量の増加にともない，日本では1958年のHIPAC 1*8を皮切りに，データを磁気テープにとっておくなんてことも行われるようになりました。このように周辺装置が増えるにつれオペレータの負担が大きくなってきました。まずプログラムの実行に必要なテープを装置にセット，プログラムを紙テープ読み取り装置にかけ，コンピュータを起動，実行が終了すれば紙テープ，磁気テープをしまい，次のプログラムのために同じ作業を操り返す（**図1**）。プリンタを使えば，それにともなう作業も必要です。

ところでこの頃のコンピュータは非常に高価なものでした。ですから有効にコンピュータを使用しないと損，なるべくむだな時間は省きたい。となればプログラムの実行が終わってから次を始めるまでの時間は短ければ短いほどいいわけです。ということから人間（オペレータ）の介入を減らし，コンピュータが連続的に処理できるようにするプログラムが作られました。これがOSの最初です。これはIBM704のFORTRANモニタシステム*9として1955年に発表されました。これで1つのプログラムから次のプログラム実行の時間は大幅に短縮されたのです。しかしまだむだがあります。I/Oが時間を費やしている間，CPUは止まってしまうのです。たとえば磁気テープにデータを記録しているときなど，データを記録し終わるまでコンピュータは何もせず待ち続けます。一般にI/Oの動作はコンピュータに比べ非常に遅いものですから，この時間は無視できないほど大きいのです。この様子が**図2(A)**に示されています。そこで，**図2(B)**のようにすればCPUの空き時間はなくなります。この図からわかるとおり，プログラム1でI/O待ちになりCPUがヒマになるとプログラム2が動きます。プログラム2がI/O待ちになればプログラム3へ……という具合で，I/O処理の待ち時間を利用して，複数のプログラムを実行するわけです。これが多重プログラミングと呼ばれるテクニックで1960年に発表されたUNIVAC IIIで初めて実用化されました。ここで述べた方法はバッチ方式といわれます。この他の多重プログラミングテクニックに実時間方式(real time)やタイムシェアリング方式(time sharing)があります。

このように大型コンピュータのOSはとてつもなく高価なコンピュータを有効に活用するために生まれ発達してきたのです。

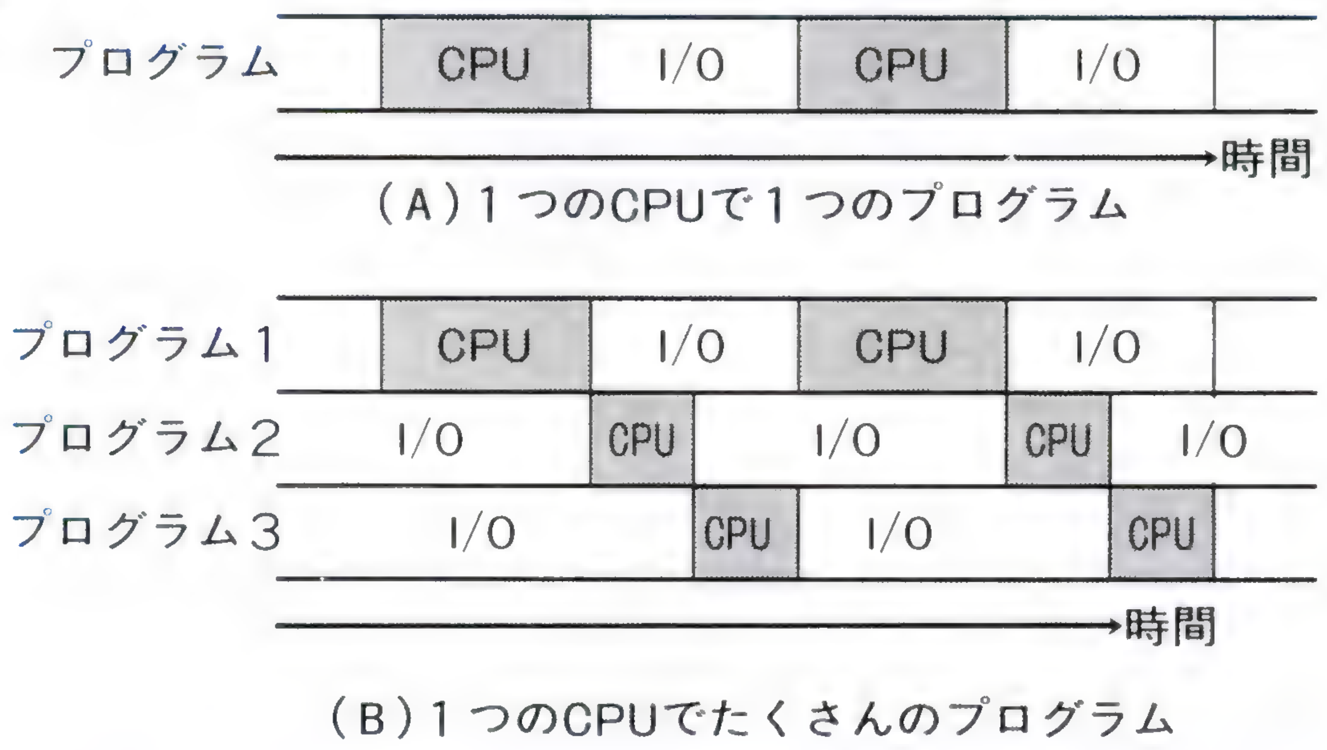
大型コンピュータとパソコンのOS

歴史のところで説明したとおり大型コンピュータのOSはコンピュータの「有効活用」のために生まれ，現在もそれが第一の存在意義になっています。「資源」(resource)ということばがあります。コンピュータ用語としては「問題を処理するのに必要なもの」として使われています。これはまあ，えらくすごい概念でして，CPU，メモリ，ディスク，プリンタ，ディスプレイ，キーボード，プログラム，データ，そしてプログラマやオペレータさえも同等に資源として取り扱ってしまうのです。現在の大型コンピュータのOSは，これらの資源をすべて有効

図1

プログラムの実行が終了すると，後しまつを人間がする。さらに準備も人間が行う。コンピュータが稼動するのは，実行時だけ。

図2



(B) 1つのCPUでたくさんのプログラム

に使うようにできています。たとえば，プログラマ，この資源を有効に使うためには，使いやすい言語を提供し，扱いやすいエディタを，そして効率よくデバッグを進められるような道具を与えなければいけません。この資源を有効に使うことによってプログラムという資源が速く完成するでしょうし，プログラム資源のコストが低下するのです。

さて，パソコンのことを考えてみましょう。パソコンは低価格で，使用コストも非常に安価です。「資源」を有効に利用することやコストは，大型機のような問題にはなりません。といっても人間が使う道具として考えれば，それが趣味で使われるとしても使いやすいほうがいいのに決まっています。そこでパソコンのOSは，「パソコンを人間（ユーザ）にとって使いやすいものにする」ことを目的としています*10。

すなわち，目的において大型機とパソコンのOSはまったく異なります。しかし「ユーザにとって使いやすい環境を作る」というのは「人的資源を有効に活用する」と同じ意味であるということができ，現在，パソコンのOSは大型機（というよりもむしろミニコン）のOSに近づいているのです。

だれのための使いやすさか

ズバリ，「プログラマのため」です。OS上で動くソフトウェアを使いやすくするため，というよりもOS上でプログラムを組みやすくするための便利さなのです。どんな言語を使うにもエディタは同一のものですみます。ディスクに対する基本的な操作は，どんなプログラムで作られたファイルにしろ共通です。ディスクやキーボードに対する入出力は，アセンブラでプログラムを作るときには後述する「スーパバイザコール（SVC）」を使えば一発ですみます。FORTRANで作ったデータファイルをCで読むようなこともできます。さらに進んだOSではリダイレクション*11，フィルタ*12，ディレクトリの階層化などもあります。これなどは使い方さえ覚えればプログラマ以外の人でも便利なんでしょうが，使い方を覚えるのがたいへんでしょう。

すなわち，OSの特長を最大限に享受できるのは，プログラムを組む人自身なのです。

パソコンのDOS

FMシリーズでディスクをつけている状態ではディスクBASICを走らせるのが普通です。このときのフリーエリアは26Kバイトほどで，通常はこれで充分だといえます。しかし他の言語を走らせるとき，26Kバイトの中ですべてを行うのは不可能です。BASICの翻訳部分など，そのときには不必要です。そこでディスクとの入出力部分だけをメモリ上にもっていれば充分です。プログラムのディスクへの出し入れは確かにこれで充分なんですが，もう少しおまけをつけてプリンタやキーボード・ディスプレイを使うのに必要最低限のものを持たせます。また，ディスクの状態を監視し，基本的な命令も持たせればよいでしょう。でもこれらを一気にメモリに入れると残るメモリが少なくなり非効率です。常時メモリに持っている必要のないものはディスクに入れておきます。これらメモリに入っている部分，そしてディスクに入っている部分の総称が「DOS」です。このようなDOSとして有名なのがCP/M，FLEX，UCSD-p Systemなどです。これをもう少し拡張してディスクに入っているコマンドが豊富になるとMS-DOS。これには階層ディレクトリがついています。さらにメモリ管理をしてマルチタスク，マルチユーザなんてのがつくと，さらに使い勝手がよくなります。OS-9，MP/M，コンカレントCP/M-86などがこれに，さらに開発ツールが加わり（これが重要です），コンソールコマンドを言語に近づくまで発達させればUNIXになります。

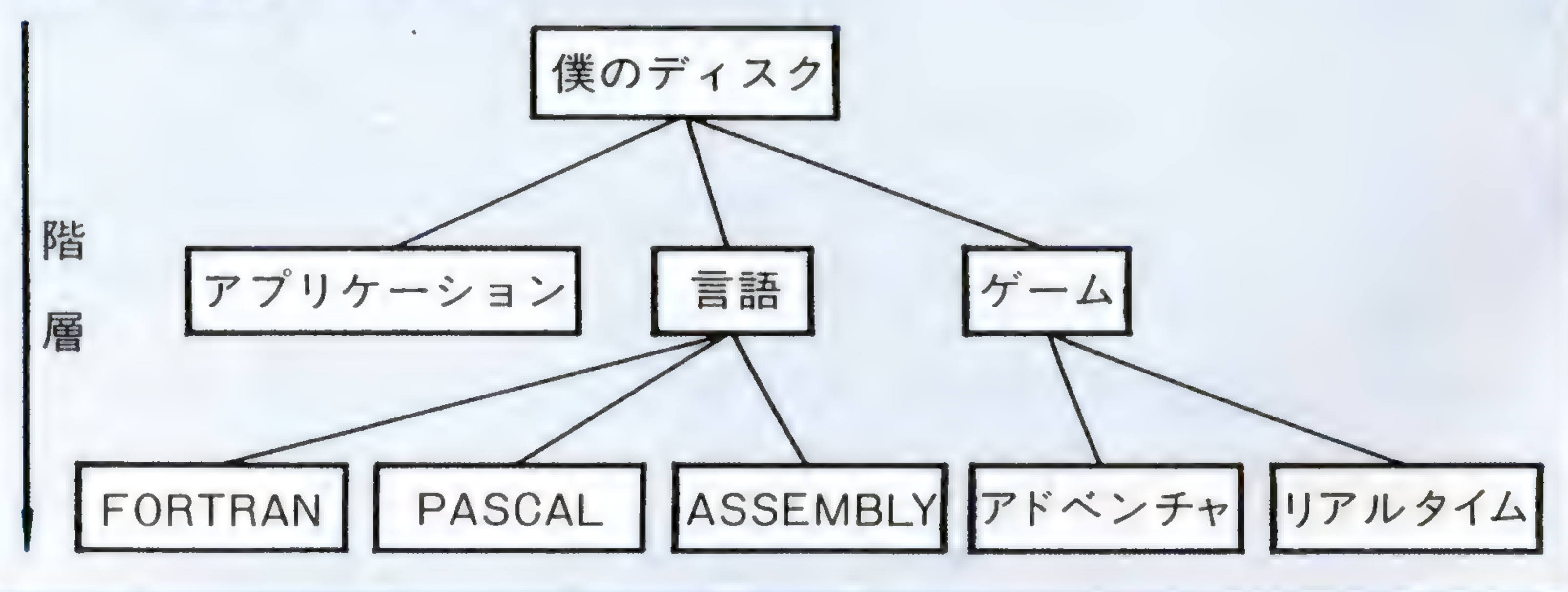
DOSの進化

前述のとおりDOSはどんどん機能が強化されてOSという名称にふさわしいものに進化していきます。そこで進化のポイントについて見ていくことにしましょう。

階層ディレクトリ

ディレクトリとはファイルを表す辞書のこと。ディスクにどんなファイルが存在するとかそのファイルがどこにあるとか，こういった情報を持っています。BASICでは“FILES”ですが，CP/M，MS-DOSでは“dir”で，FLEXでは“CAT”あるいは“DSP”で見ることができます。BASICのFILES命令ではすべてのファイル名が出てきますが，ファイルが増えてくると多すぎて何が何やらわからなくなってしまいます。容量の小さい補助記憶装置を使っていれば問題はないのですが，8インチの両面倍密ディスク，ハードディスクなど1つの装置で1Mバイトを超えるようになると，この問題が生じてきます。しかしファイルをグループ分けしておけば混乱したりはしません。同様なファイルをグルーピングし，レベルづけをした構造が階層ディレクトリです。階層ディレクトリの例を図3に示しておきます。

図3　階層ディレクトリ



また，ディレクトリさえ異なれば，同一のファイルネームを使ってもだいじょうぶです。何人かで1つの記憶装置を共有するような場合，必要不可欠な機能といえるでしょう。

マルチタスク，マルチユーザ

OSの歴史のところで書いた多重プログラミングでの用語です。タスクとは独立したプログラムのことで，同時に何本ものプログラムが走るということ。マルチユーザは同時に(ここでいっているのは「見かけ上」の同時性)複数のユーザがコンピュータを使えること。マルチタスクの場合，前の説明のようにCPUが空いている時間だけ別の仕事をすればいいのですが，マルチユーザの場合そうはいきません。たとえばAというユーザがI/Oをいっさい必要とせず延々とCPUを使い続けると，別のユーザBは何もできなくなってしまいます。そのために一定時間ごとにCPUが処理する対象，つまりユーザを切り換えてやらなければいけません。マルチユーザを実現するためにはシステムがタイマ割り込み機構を持つ必要があるのです。

マルチタスクより低次元にスプールという機能があります。これはSimultaneous Peripheral Operation On-Lineの略で周辺機器同時動作などと申します。これは特定の周辺機器についての機能なので，マルチタスク機能の一部分とみなせます。FLEXはシングルタスクのOSですが，プリンタに対するスプール機能は持っています。

スーパバイザ，スーパバイザコール(SVC)

DOSのうちメモリに常駐するプログラムでプログラム管理(前述のマルチプログラミングに出てきたバッチとかタイムシェアリングなどの制御)や入出力を行う部分をスーパバイザ(supervisor)と呼びます。CP/MなどのDOSではたいへん小さなものですが，マルチタスク，マルチユーザの機能を提供するDOSではかなり大きくなっています。

このスーパバイザがユーザに対して基本的な入出力ルーチンを提供します。このルーチンを呼び出し使用することをSVC (Super Visor Call)と呼び，またの名をファンクションコール，はたまたシステムコールなどと称します。これにはコンソール(ディスプレイ)に対する文字出力や，コンソール(キーボード)からの文字入力，プリンタへの出力，ディスクへの入出力，さらに高級なDOSになると階層ディレクトリや，タスクの登録などマルチプログラミングに対する機能が含まれます。これら基本的な入出力をSVCで行うために機種，すなわちハードウェアの差がソフトウェア側から見ればなくなり，よって機種が違ってもDOSが同じであれば同じソフトウェアが使えるのです。また，このことから，スーパバイザは機種により変更する必要があるということもおわかりでしょう。

プリヴィリッジドモード(特権モード)

一般にシステムが大きくなればなるほど，信頼性を上げる必要があります。というのもシステムが大きくなり複数のユーザが同一のシステムを使うようになると，だれかのプログラムが他人のデータを盗んだり誤って破壊したりする可能性が出てくるからです。そのため，メモリの保護機能が必要となりますが，これをユーザが勝手にいじると何にもなりません。そこでメモリの保護機能は特権を持ったもの(OS)しか持てないようにします。このような特権を持つのが，前述のスーパバイザです。 80286, 68000, Z8000はこのモードを持っていますが，8086にはありません。

最後に

OSを知ることはマニアにとっては非常に興味深くおもしろいことですが，一般の「使う人」であるユーザにとってはどうなのでしょう。統合化ソフトウェアという方式があり，これがあれば特に専門知識がなくても(多くの場合)必要とされるソフトを使うことができます。が，OSもこの方式に近づきつつあり，MS-Windowsなど，その例といえましょう。これはこれとして，プログラマにとってみればUNIXのような居心地のよいソフトウェア環境がパソコンで実現されることが望ましいでしょう。今後もOSは，さらに進化していくことと思われます。

参考文献

1：小田徹「コンピュータ史」(オーム社)
2：高橋・土居・益田「オペレーティング・システムの機能と構成」(岩波書店)
3：江村潤朗「オペレーティングシステム入門」(オーム社)
4：Mark Dahmke著　玄光男・中原紀共訳「マイコンOS」(共立出版)
5：アンソニーチャンダー著　坂井利之訳「コンピュータ用語辞典」(講談社)
6：渡辺茂監修「マイコン小事典」(講談社)

用語説明

＊1) TOS (Tape Operating System)というのもある。
＊2) 言語を解釈し，機械語に翻訳し，その結果をディスクにセーブする。
＊3) 翻訳された機械語プログラムは，多くの場合，そのまま実行できない。リンクすることにより実行可能な形式となる。リンクは，翻訳ずみのプログラムを連結する。
＊4) スーパバイザがサポートしていないということ。スーパバイザについては後述。
＊5) メモリに記憶されたプログラムをCPUに読み出し，逐次，実行する。
＊6) レミントンランド社。
＊7) 磁気ディスクはかなり遅れて1958年IBM 305 RAMACで最初に使われた。
＊8) 基本素子はパラメトロン。
＊9) 開発はゼネラルモータース社，ノースアメリカンエィヴィエーション社。
＊10) ミニコンのOSも，これを目的としている。
＊11) 対象とするI/Oを自由に変更できる機能。
＊12) パイプという機能により，プログラムの出力を別のプログラムをとおして得られる。これをフィルタという。

こんなにある FMで使えるOS

——各OS紹介

兼子　毅

　昭和56年初夏，FMシリーズ第1号である「FM-8」が発表されました（当時はマイクロ・エイトと呼んでいた）。漢字ROMを使っての日本語表示や640×200ドットのフルカラーグラフィックなど，今では当たり前になってしまいましたが発売当時は画期的なものでした。もう1つFM-8の特徴として（話題のバブルカセットを補助記憶装置として採用したこともそうですが），3種の流通DOS，つまりUCSD p-System，FLEX，そしてCP/Mを利用できることがあげられます。57年8月には星光電子からOS-9が発表され，富士通からは8088カードがCP/M-86付きで提供されるようになり，FM-8上で走るOSは全部で5種類になりました。このことはパソコン用として使われる汎用OSのすべてが使える，ということとほとんど同じ意味です。FM-7とFM-11が発表されたのは57年の暮れごろですが，やはり各種のOSが走るようになっています。

　というわけでFMシリーズで使えるOSを紹介するということは，パソコン用のOSを紹介することになってしまいます。ところが成り立ちや設計思想，CPUも異なるOSを同列に並べるのは至難の技です。でも頑張って書くことにします。読み終わって少しでも面白かったとか，ためになったとか思った人は，ぜひ編集部気付ではげましの手紙を出しましょう。女性（ただし20歳以下）にはちゃんと返事を書きますから。

　まずは**表1**を見て，FMシリーズで走るOSを把握してください。

表1

OS名	機　　種	メディア	動作CPU	価　格（円）
OS-9 L1	7,8,11EX・AD・ST	5D	6809	48,000
L1	77	3.5D	6809	未　定
L2	11EX・AD・ST	5D ,8D	6809	58,000
L2	11AD2	5HD	6809	標準添付
L2	77	3.5D	6809	未定（400ラインセット要）
FLEX	7,8,11EX・AD・ST	5D ,8D *1	6809	48,000
CP/M	7,8,11EX・AD・ST	5D,8D *1	Z 80	Z 80カードに添付
	77	3.5D	Z 80	Z 80カードに添付
CP/M-86	8,11EX・AD・ST	5D,8D	8088	8088カードに添付 *2
CP/M-86	11BS	5HD	8088	標準添付
MS-DOS	11EX・AD・ST	5D,8D	8088	43,000
MS-DOS	11BS	5HD	8088	43,000
UCSD pV2.0	7,8,11EX・AD・ST	5D	6809	98,000
UCSD pV4.1	11EX・AD・ST	5D	8088	146,000

注）5Dは5インチ両面倍密。8Dは8インチ両面倍密。5HDは5インチ両面高密度倍トラック。3.5Dは3.5インチ両面倍密。7はNEW7を含む。また，適応機種は標準的なドライブシステムを基準としている。したがって，たとえば77に5インチドライブを増設した場合は，7用が使える。

＊1　7用の8D版はない。

＊2　11EXは標準添付。

各OSの略史と概要

▶OS-9

　モトローラ社が6800を発表したのが昭和49年のことです。その後も改良が進み，「究極の8ビット」と呼ばれる6809が発表されたのが昭和54年です。6809 CPUの開発と並行して，6809の能力をフルに引き出すBASIC言語の開発が，昭和53年から55年にかけて行われました。マイクロウェア社との共同開発による「BASIC09」がそれです。このBASIC09をサポートするOSを，ということでマイクロウェア社によって開発されたのが「OS-9」です。

　日本では，昭和57年に8用のOS-9 V1.1が発売され，翌年にはV1.2にバージョンアップされ，11用のレベル2も発売されました。59年には，日本語機能を強化したV1.2Jも発表されました。

　OS-9の特徴は，何といっても「UNIXライクなマルチユーザ・マルチタスクのOSである」ということでしょう。さらに，メモリモジュールの概念の導入による，ソフトウェア開発環境の向上もあげられるでしょう。

▶CP/M

　昭和48年ごろ，当時インテル社のコンサルタントをしていたG. Kildall（現デジタル・リサーチ社社長）が，その原形を開発しました。その後51年になって，自身でデジタル・リサーチ社を設立し，CP/M V1.4を発表しました。当時，マイコンのCPUは80系が支配的で（今でもそうですが…残念ながら），80系のためのOSであるCP/Mも大変普及しました。54年にはV2.0に，翌55年には現在のV2.2になりました。FMシリーズのZ-80カードで走るのはこのバージョンです。さらに58年には，8ビット用CP/Mの最終バージョンである（とデジタル・リサーチ社は言っています）「CP/Mプラス（CP/M V3.0）」が発表されました。しかしおそらくCP/Mプラスは，FMシリーズ上で走ることはないと思います。

　16ビットCPUの8086/8088用に開発されたものがCP/M-86です。

　CP/Mの特徴は，構造のシンプルさと，利用できるソフトウェアの多さではないでしょうか。

▶FLEX

FLEXはテクニカル・システムズ・コンサルタンツ（TSC）社が開発した68系のOSです。68系のCPUは80系と異なり，コードレベルでのコンパチビリティはないので，6800FLEXと6809FLEXの２種類があります。実は私の手元には，FLEXの開発経緯などを書いた資料がまったくないので，詳しいことはよくわかりません（ごめんなさい)。ただ，FLEXのシステムデザインが明らかにCP/Mを意識して作られているので，6800FLEXが開発されたのが昭和53年ごろ,6809FLEXは55年ごろに開発されたと思われます。FMシリーズで走る6809FLEXは，V3.××（マイナーチェンジがしばしば行われているので，購入した時期によって，××は異なる)ですが，今年(59年)になって，FM-7/8用に「日本語FLEX V4.01」が発表されました。

FLEXの特徴を一言で言えば，6809上で走るCP/MライクなOSというところでしょうか。

▶UCSD p-System

p-System はその名のとおり，カリフォルニア大学サンディエゴ分校（UCSD）で開発されました。もともとPascal言語をミニコン上で走らせるためのもので，その後マイコン用に移植されました（VI.4）。その後，ソフテック・マイクロシステムズ社が，ライセンスを獲得するのとほぼ同時に，VII.0が発表されました。Apple II上で走るApple Pascalも実は，p-System VII.1のことでした。昭和55年になってVIV.0になり現在に至っています。FMシリーズで走っているp-Systemは，VIV.1です。

p-Systemの特徴は，Pascalで書かれた，マシンインディペンデントなOSということでしょう。

▶MS-DOS

大型計算機の世界では巨人といわれているIBMが，パソコンの世界に殴り込みをかけたときにOSとして採用したのがMS-DOSです。はじめIBMは，すでにCP/M-86などを開発していたデジタル・リサーチ社に依頼したそうですが，なぜか断られてしまいました。そこでマイクロソフト社に開発を依頼しました。マイクロソフト社は，あるソフト会社が開発した8086／8088用のOSに目をつけ，ライセンスを買い上げました。これがMS-DOS誕生の経緯です。昭和56年に出たV1.25は，ほとんどCP/Mと同じでしたが，58年には，V2.0を発表しました。これはUNIXの味つけがされていて，かなり使い勝手が良くなっています。11で走るMS-DOSは最新バージョンのV2.11です。

各OSの内部構造

各OSがどのようにメモリや周辺機器を管理しているか，ということを知るために，それぞれのOSの内部構造について簡単に調べておきましょう。説明の都合上，まずCP/M-80から……。

▶CP/M-80

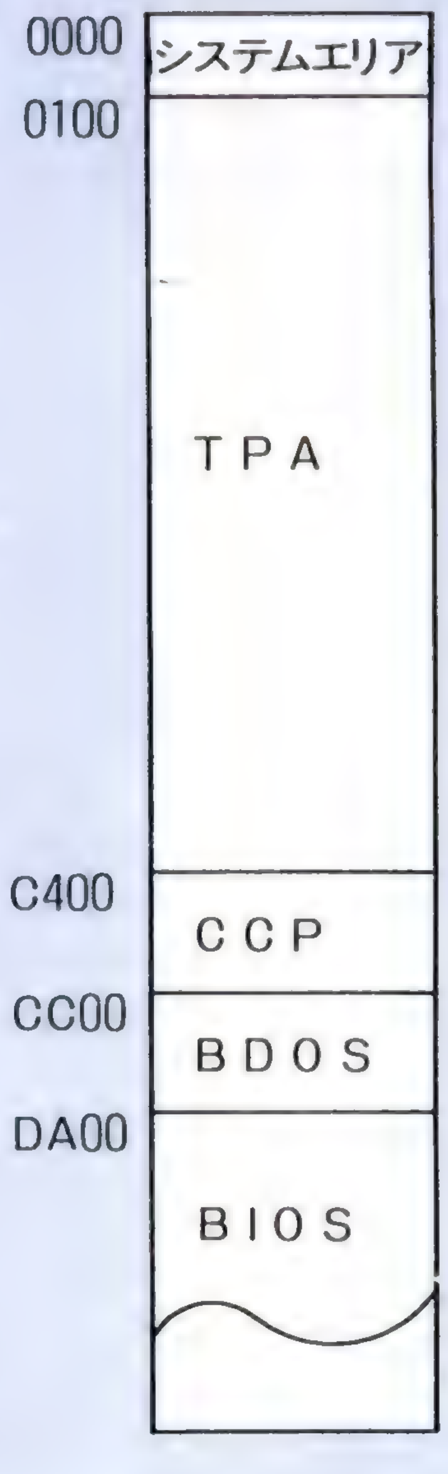
CP/Mのシステムディスクをドライブ0に入れて電源スイッチを入れると，ドライブがアクセスされて2～3秒でCP/Mが起動します。そのときのメインシステムのメモリマップは図1のようになっています。上から順に見ていきましょう。0000番地から00FF番地までは,「システムスクラッチエリア」と呼ばれ，CP/Mシステムが使うワークエリアです。このエリアは，どの番地を何に使うということがすべてのCP/Mについて共通に決められていて，CP/Mシステムの汎用性を支える一つの理由でもあります。0100番地からC3FF番地までは,「トランジェントプログラムエリア（TPA)」と呼ばれていて，要するにプログラム用のフリーエリアです。C400番地からCBFF番地までは「コンソールコマンドプロセッサ（CCP)」，CC00番地からD9FF番地までが「ベーシックディスクオペレーティングシステム（BDOS)」，DA00番地以降が「ベーシックインプット・アウトプットシステム(BIOS)」と呼ばれています。

BIOSは，15個のサブルーチンと２個のブートベクトルを持つ，CP/Mシステムの中で唯一ハードウェアに依存する部分です。BIOSには，ディスクの１セクタリード／ライトや，コンソール入出力といったもっとも基本的なサブルーチンが含まれています（表２参照)。

BDOSは，７個のサブルーチンを持つ，ディスクドライブの管理部で，CP/Mのもっとも重要な部分です(表３参照)。

CCPは，ユーザがキーボードから打ち込

図１ CP/M-80メモリマップ



んだ文字列を読み取り，コマンドファイルをメモリにロードすることをBDOSに依頼したり，コマンドに渡すパラメータをセットしたりします。

▶CP/M-86

CP/M-86 の内部構造は，基本的には前述のCP/M-80 と同じです。BDOS, BIOSともに拡張されています（図2参照）。また，メモリ管理方式も改善され，最大8個のプログラムをロードすることができますが，それらが同時に実行されるわけではありません。

表2　CP/M　BIOS

BOOT	コールドスタート
WBOOT	ウォームスタート
CONST	コンソールREADYチェック
CONIN	コンソールからの読み込み
CONOUT	コンソール出力
LIST	プリンタ出力
PUNCH	パンチャ出力
READER	リーダからの入力
HOME	シークトラック0
SELDSK	ディスクドライブ選択
SETTRK	トラック指定
SETSEC	セクタ指定
SETDMA	DMAアドレス指定
READ	1セクタリード
WRITE	1セクタライト
LISTST	リスト状態に戻る
SECTRAN	セクタ変換サブルーチン

▶MS-DOS

メモリマップは図3のとおりです。CP/M-86とほぼ同じ構造で，COMMAND.COMとCCPが，MSDOS.SYSとBDOSが，IO.SYSとBIOSがそれぞれ対応し，機能も対応しています。ただし，CP/M-86 に比べMS-DOS の方が，それぞれ高い機能を持っています。とくに，V2.11ではUNIXを意識した機能拡張が行われ，階層化ディレクトリや，パイプラインなどもサポートしています。

表3 CP/M　BDOS

SEARCH	：名前でファイルを探す
OPEN	：ファイルのオープン
CLOSE	：ファイルのクローズ
RENAME	：ファイルの変更
READ	：ファイルのリード
WRITE	：ファイルのライト
SELECT	：ディスクドライブの選択

▶FLEX

メモリマップは図4のとおりです。他のOSに比べると多少ごちゃごちゃした感じがしますが，基本的にはDOS（ディスクオペレーティングシステム）とFMS（ファイルマネジメントシステム）に分けられます。DOSの部分には，基本的なサブルーチンがあり，FMS は文字どおり，外部記憶装置上のファイルを管理する部分です。

働き

今まで簡単に説明した4種のOSはどれもシングルユーザ・シングルタスクのオペレーティングシステムです。これらのOSの基本的な機能は，外部記憶装置（ディスクなど）上にあるファイルを名前で管理する，ということです。ファイルの概念はちょっと難しいのですが，一言で言えばデータの集まりです。機械語のプログラムかもしれないし，BASIC のテキストかもしれ

図2　CP/M-86メモリマップ

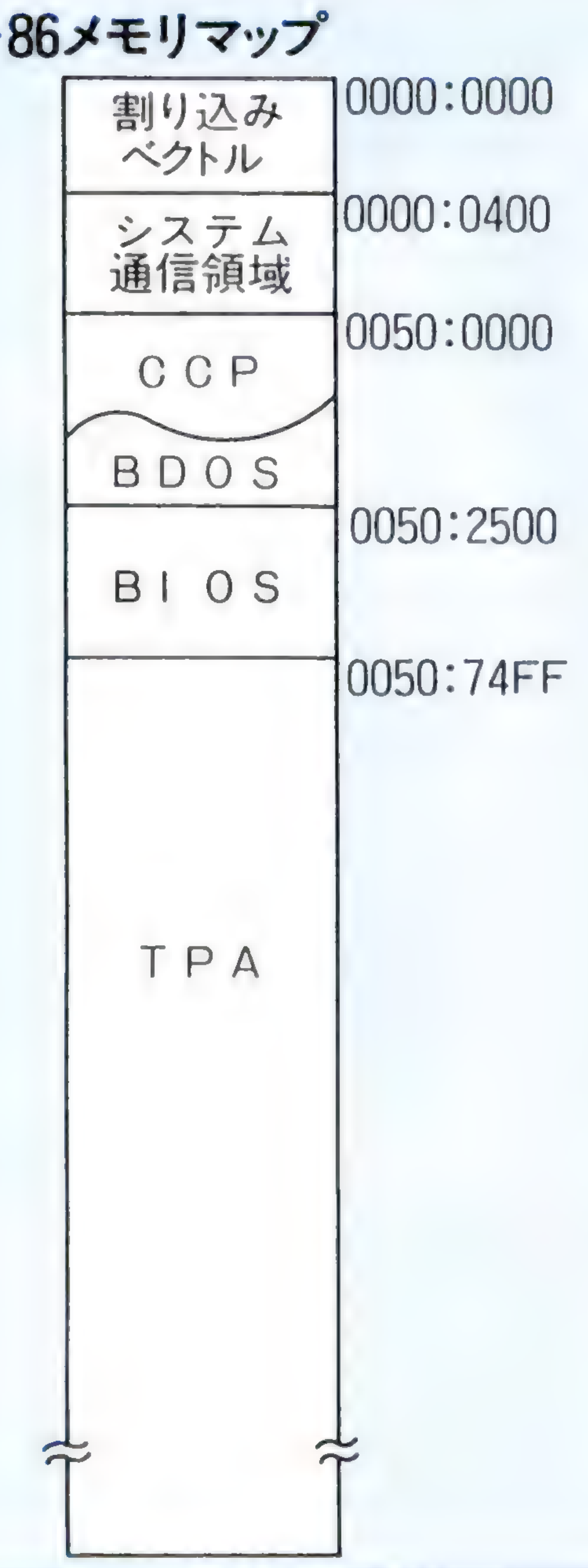


図3　MS-DOSメモリマップ

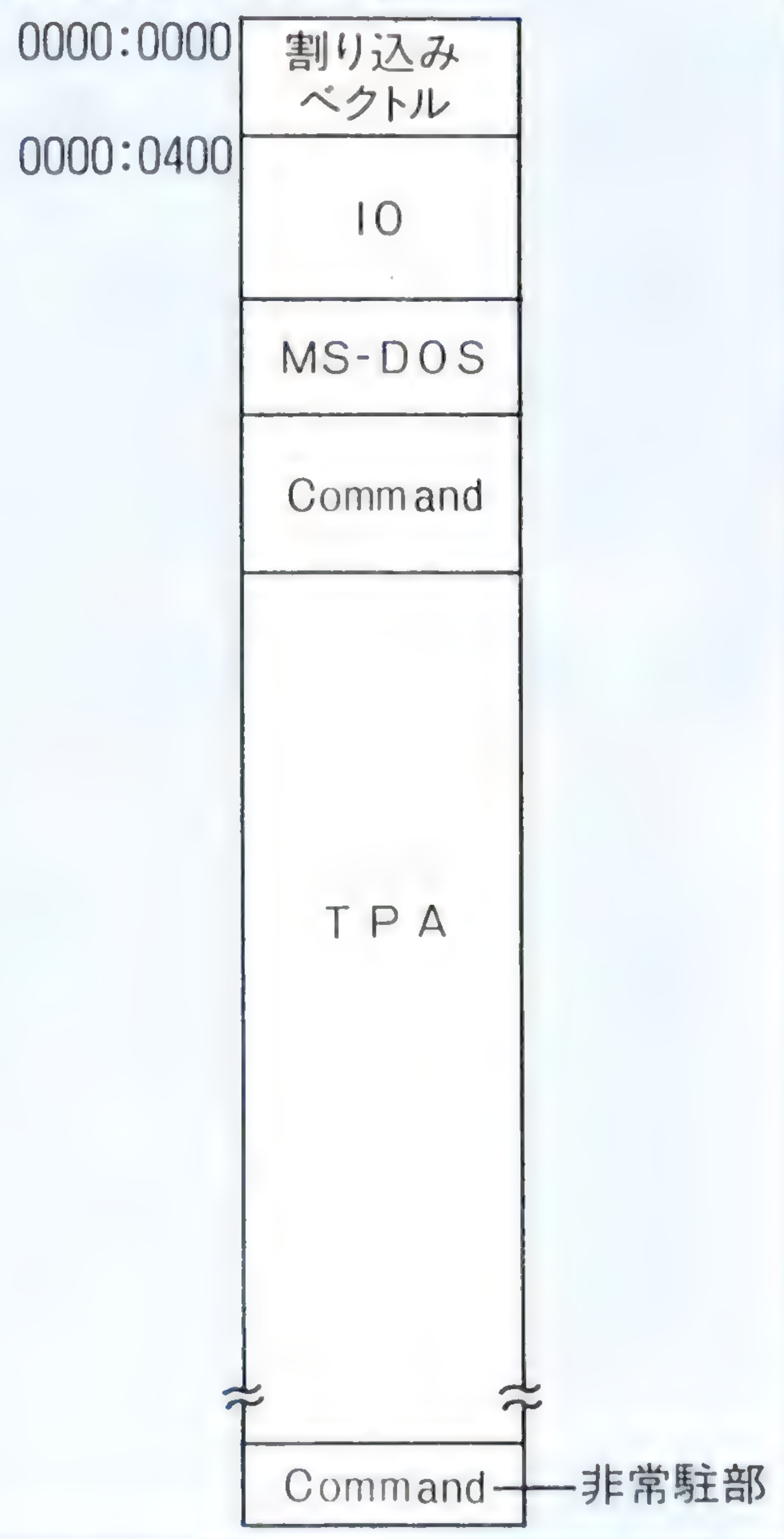
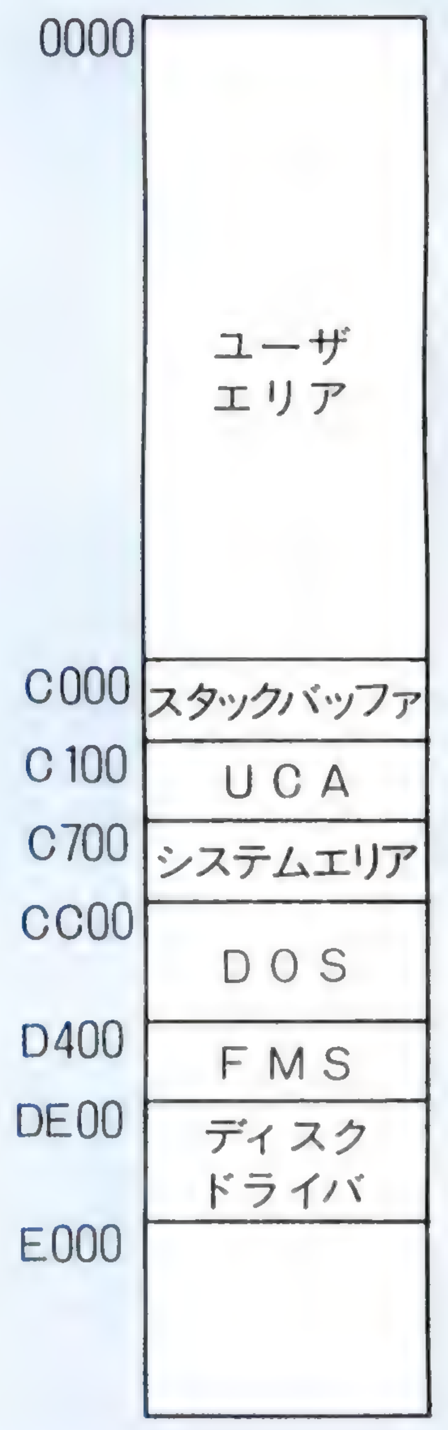


図4　FLEXメモリマップ



ないし，日本語ワープロで作成した文書のこともあるでしょう。このうち，メモリ上にロードして実行することのできるプログラム（普通は機械語）をコマンドファイルと呼びます。

さて，ファイル名のつけ方はそれぞれのOSによっていくつかの制約があります。前述の4種のOS共通の約束として拡張子（エクステンション）があります。これは，田中先生，山田部長などの肩書きのようなもので，そのファイルがどんな種類かを表しています。拡張子もファイル名の1部ですから自由につけていいのですが，システムで決められているものも何種類かあって，コマンドファイルにはCOMとかCMDとかをつけることになっています。

では，上に述べた4種のOSがどんな働きをするのか簡単に見てみましょう。ディスク上にGREETというコマンドファイルがあり，これを実行させることにします。

1）キーボードからGREETと打ち込みリターンキーを押す。
2）GREETというコマンドファイルをディスクから探して，もし存在すればメモリ上にロードする。
3）ロードされたプログラムにCPUの制御を移す。

これでおしまいです。CP/MでもFLEXでもMS-DOSでも，働きはこれだけのことです。

▶OS-9

OS-9は今まで述べてきたシングルタスクのOSとはまったく設計思想が異なります。カーネルと呼ばれる部分がOS-9の中心的な部分ですが，ここの働きは，マルチタスクで並列して走っているプロセスの管理です。図7に書いてあるようにシングルタスクのOSでは，プログラムをメモリ上にロードして実行してしまえばいいだけですが，マルチタスクのOSでは，タスクの切り換えを間違いなく行わねばなりません。ディスク上のファイルの管理などは，ファイルマネージャという部分が担当します。

図7　シングルタスクとマルチタスク

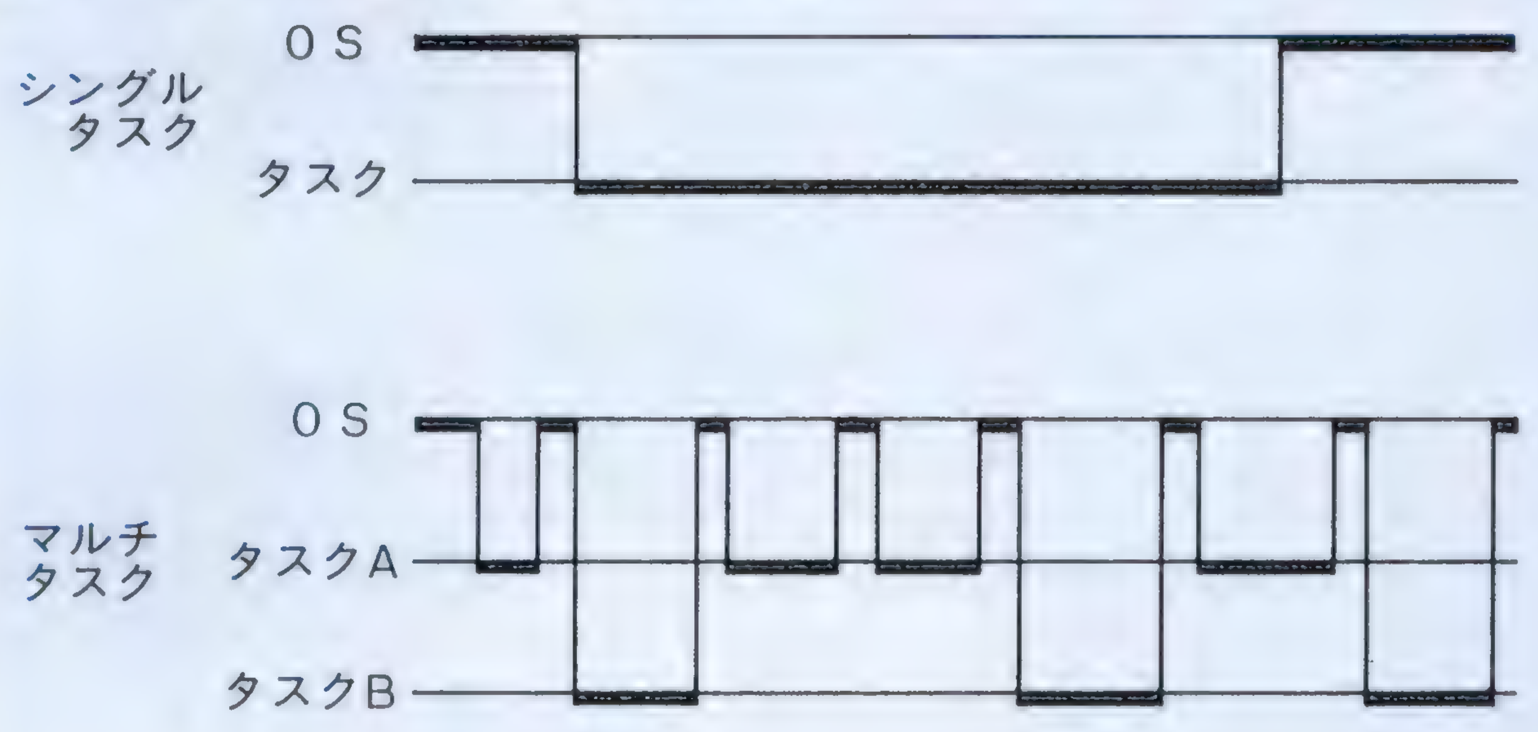
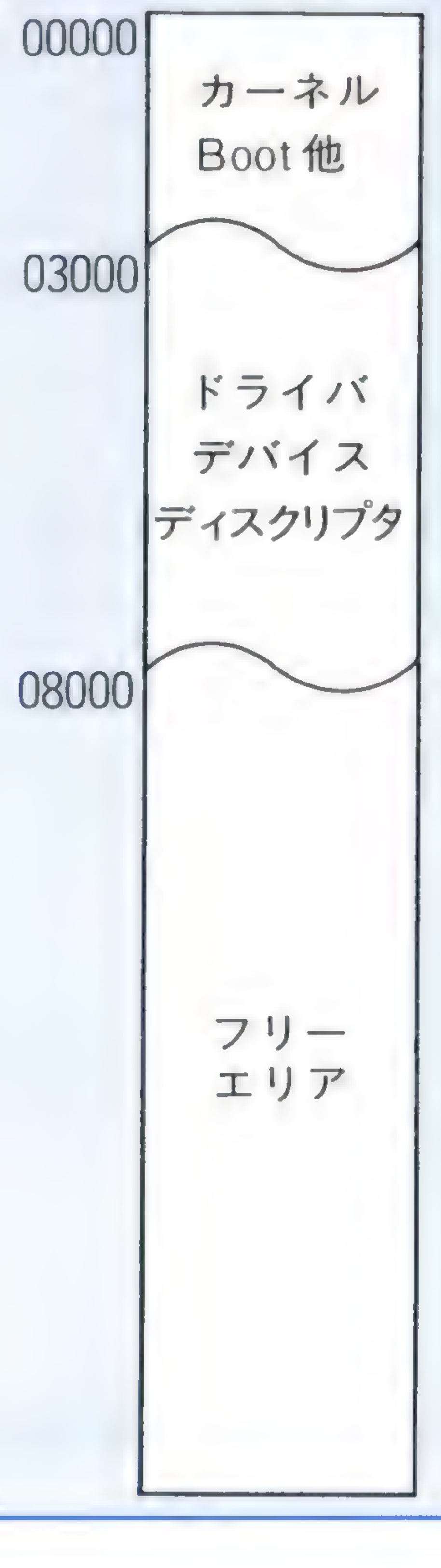


図5　OS-9 L1メモリマップ

図6　OS-9 L2メモリマップ



▶UCSD p-System

p-Systemは，シングルユーザ・シングルタスクのOSですから，基本線は他のシングルタスクOSと同じです。ただ，メニュ形式（実行できるコマンドが，メニュ形式で画面に表示され，一文字だけキー入力すればよい）になっているため，多少様子は異なります。p-Systemは，pコードと呼ばれる仮想的な機械語で書かれています。この機械語をそのままFMシリーズのCPUで実行することはできないので，6809や8086の機械語で書かれたpコードインタプリタがpコードを解釈しながら実行していきます。SYSTEM.PASCALがOSの本体でSYSTEM.INTERPがpコードのインタプリタです。これらが通常メモリ上に常駐しています。

コマンド

各OSのパッケージを購入すると，ディスクが数枚とマニュアルが数冊はいっています。ディスクには，OS本体と多くのユーティリティコマンドが入っています。これらのコマンドは，OSの一部というわけではありませんが，それらのコマンド（プログラム）がなければ，せっかくのOSも使いにくくなったり，事実上使えなくなっ

たりします（たとえば，ディスクをフォーマットするユーティリティがないことを想像してください。新品のディスクをフォーマットできなければ，結局その OS を使うことは不可能です)。各コマンドの一覧を**表4**に示しました。だいたいどの OS も似たようなコマンドを用意してあります。その中で，いくつかを比較してみましょう。

▶エディタ

ある調査によると，コンピュータを使用している時間の99%以上は，ワードプロセッシングに費やされているそうです。もちろん，英文ワープロや日本語ワープロとして使っているときもそうだし，プログラムを打ち込むときもそうです。つまり，エディタの使い勝手の良し悪しが作業の能率に大きな影響を与えるというわけです。エディタの働きを大きく2つに分けると，

1）BASIC やアセンブラなどの，プログラムテキストを作成する。

2）手紙やマニュアルなど，文章を作成する。

ということになります。前者を「エディット」，後者を「ワードプロセッシング」と呼ぶことが多いようです。一般に OS に付属しているエディタは，(1)の機能しかありません。もちろん，手紙などの文章を作成することも不可能ではありませんが。エディタの分類の仕方で，

1）ラインエディタ

2）スクリーンエディタ

という分け方があります。F-BASIC のエディタなどは，スクリーンエディタです。

CP/MのED.COM(.CMD)

テキスト作成用のラインエディタです。ごく普通のエディタですが，割と使いやすいと思います。

OS-9のEdit

やはり，テキスト作成用のラインエディタです。機能は豊富ですが，やや使い勝手が悪いようです。

FLEXのEDIT.CMD

テキスト作成用のラインエディタです。TABストップの位置を自由に変えられるなどの機能もあります。

なお7／8用の日本語FLEXは通常は日本語モードですが，英数字モードに切り換えることができ，そのときはカーソルキーやDUPキーなどを用いたスクリーンエディットができます。ただしこれは，本格的なスクリーンエディタが搭載されているという意味ではなく，サブシステムの機能を使っているためいろいろ制約があります。

MS-DOSのEDLIN.COM

文字どおり，テキスト作成用のラインエディタです。CP/MのEDを意識して作られたようで，サブコマンドも多くが共通です。

p-SystemのEditor

文章作成もできるスクリーンエディタです。このエディタは大変使いやすいという定評があります。ちゃんとしたスクリーンエディタを標準で持っている OS はp-Systemだけです。また，以前のバージョンで

表4

	機能	CP/M	CP/M-86	FLEX	OS-9	UCSD p-System	MS-DOS
ファイル関係	ディレクトリ表示	dir stat	dir stat	cat dsp	dir	F) E) L)	dir
	ファイル削除	era	era	delete xout	del	F) R)	del erase
	ASCII ファイル表示	type	type	list	list		type
	ファイル名変更	ren	ren	rename	rename	F) C)	ren rename
	ファイル属性変更	stat	stat	prot	Attr		
	ファイルコピー	pip	pip	copy	copy	F) T)	copy
階層ディレクトリ関係	ディレクトリ変更				CHD chx		chdir cd
	カレントディレクトリ表				pwd pwx		chdir cd
	ディレクトリ作成				makdir		makdir md
	ディレクトリ削除				deldir		rmdir rd
ディスク関係	フォーマット	format	format	rd	format	X) formatter	format
	ディスクコピー	(sdcopy)	copydisk		backup	F) T)	diskcopy
	ディスクのチェック				dcheck free	F) B)	chdisk
	ディスクの修復					F) X)	recover
	システムディスク関係	sysgen	Ldcopy	link	os9gen putldr	X) booter	sys
プリンタ関係	プリントスプーリング		print	print Qcheck			print
漢字関係	漢字モードへの切り換え	selcode	selcode	キー入力による切り換え	kmode kset		
	プリンタサポート		prtmode usrprt	システムに組み込み	pmode		システムに組み込み
開発環境	アセンブラ	ASM	ASM86	asmb	asm	A)	masm
	エディタ	ed	ed	edit build	edit build	E)	edlin
	デバッガ	ddt	ddt86		debug		debug
	ローダなど	load	gencmd				link exe2lin
	高級言語				BASIC09	Pascal	

標準のエディタであった，YALOE というラインエディタも付いています。

▶アセンブラ

OS を使いたい理由の1つに，機械語でプログラムを開発したいということがあると思います。また，もともと OS は，機械語プログラム開発支援用に作られてもいます。

CP/MのASM.COM

CP/M-80は，インテル8080用の，また，CP/M-86は，インテル8086用のアセンブラを装備しています。80系の CPU について知っている人はわかると思いますが，インテル系・ザイログ系の2種類のニモニックがあり，CP/Mで採用しているのは，インテル系の方です。8086用のASM 86 の方には，コード・マクロ機能があります。

FLEXのASMB.CMD

6809用のアセンブラです。当然マクロ機能もあります。F-BASIC 上で走る機械語プログラムの多くは，この FLEX のアセンブラで開発されたものです。

OS-9のAsm

当然6809用のアセンブラです。OS-9では，メモリモジュールという考えを導入しており，モジュールヘッダや CRC 値の計算などが必要で，このアセンブラはそれらをサポートしています。ただしマクロ機能はありません。

MS-DOSのMASM.EXE

8086用のアセンブラで，マクロ機能もあります。このアセンブラは，リロケータブルオブジェクトを出力し，リンカを用いて他のコンパイラなどで作成したプログラムと結合させたりすることもできます。

p-Systemのアセンブラ

p- System にも一応その CPU 用のアセンブラが付属しています。アセンブラで作成したプログラムを Pascal のプログラムから，外部プロシジャとして呼び出すこともできます。ただ，元来p- System は，Pascal 言語でプログラムを開発するための OS で，アセンブラ開発には向いていません。

UNIXと各OS

現在ミニコンの世界では，UNIX 系の OS が主流を占めているといわれています。マイコン用の OS も，どんどん UNIX に近づいていくと思われます。それは，UNIX が非常によくできた OS だからなのですが，OS 本体のサイズも大きく，高速の外部記憶装置（5インチや8インチのディスクは決して高速ではありません）がつながっていることが前提となっていて，そのままの形でマイコンに搭載するのは今のところ無意味です。そこで各 OS とも，その思想や特徴を導入しながら，独自の形式でUNIX風の味つけをしています。

▶CP/M,CP/M-86

FM シリーズ上で走る CP/Mファミリの OS では，まったく UNIX風の味つけはされていません。その意味では，古いタイプの OS と呼ぶことができるでしょう。

▶FLEX

入出力リダイレクトに近いことができます。たとえば，“DSP” というコマンドはディレクトリを表示するものですが，次のように入力すると，プリンタに出力されます。

P, DSP

また，次のように打ち込むと，CATALOG. OUTというファイルに，ディレクトリが出力されます。

O, CATALOG, DSP

▶OS-9

もともと UNIX を意識して作られており，入出力リダイレクトやパイプラインなどは完璧にサポートされています。

dir >/p ……プリンタへ出力

dir > catalog ……カレントデータディレクトリ中のファイルに出力

▶MS-DOS

CP/Mを意識して作られたV1.25にUNIXを意識しながら追加・拡張をしたのがV2.0です。入出力のリダイレクトやパイプラインもサポートしています。ただし，OS-9などの「正統派」パイプラインがメモリ上に展開されるのに対し，MS-DOSではディスク上に一時ファイルが作られます。

▶UCSD p-System

入出力リダイレクトをサポートしています。

日本語機能

今まで書いてきた OS は，すべてアメリカで開発されたもので，もともと英数字といくつかの記号だけ（JIS コード表の，&

表5 UNIX系の要素

	リダイレクト入出力	パイプライン	マルチタスク	
				スプーリングプリント
FLEX	可	×		○
OS-9	○	○	○	
p-System	可	×	可*	
CP/M-86	×	×		○
MS-DOS	○	○		○

*パスカルプログラムのコンカレント実行が可能

H00から&H7Fまで)をサポートしていました。今でも，CP/MやFLEX 上で走るソフトの中には，カナ文字さえ使えないものがあります。しかし，コンピュータのハード・ソフトの市場として日本が重要な位置を占めてくるようになり，漢字を使えるかどうかがポイントの一つになってきました。

▶CP/M-80

漢字を表示する機能はあります。しかし，テキスト中に漢字を書くことはできません。

▶FLEX

現在のところ，7/8用にのみ日本語FLEXが販売されています。漢字の入力は，JIS第一水準漢字一覧表にある読みを1文字入力し，「↓」キーを押すと，画面上部に対応する漢字が16個ずつ表示され，その中から選びます。辞書を用いた変換は，将来サポートされる予定です。プリンタへの漢字への印字は，GP-250F系，MP-80系，富士通ビジネスプリンタ，漢字プリンタで可能です。

表6 日本語機能の比較

	変換方式				辞書		文節変換	プリンタ出力			
	コード入力	一字検索	カナ漢字	ローマ字漢字	単漢字	熟語		GP系	MP系	ビジネス	漢字
日本語FLEX		○						○	○	○	○
OS-9V1.2	○								○	○	
OS-9V1.2J			○	○	○			○	○	○	○
CP/M-86			○	○		○			○	○	○
MS-DOS			○	○		○	○			○	○

▶OS-9

V1.2では，INSキーを押したあと，JIS漢字コードを16進で入力すると，カーソルのところにその漢字が挿入されます。V1.2Jでは，単漢字辞書による変換をサポートしています。まず，[ctrl]－[K]を押し，漢字の読みを1～5文字入力し，「→」キーを押します。するとカーソルのところに漢字が表示されます。この辞書には，最終使用文字学習機能があります。

▶CP/M-86

熟語辞書を持ち，変換することができます。辞書のメンテナンス機能もあります。

▶MS-DOS

熟語辞書を持ち，文節変換が可能です。辞書のメンテナンス機能もあります。FMシリーズのOSの中では，もっとも優れた変換機能を持っています。

画面上に漢字を出力することはできます。

ワンポイント

今まで，なるべく各OSを並列において比較してきましたが，やはりそれぞれのセールスポイントやウイークポイントはよくわからなかったのではないでしょうか。そこで，すこし無責任ですが，思い入れを込めて書いてみたいと思います。

▶CP/M-80

8ビットCPUのためのOSの中では，利用できるソフト量がダントツです。FLEXやOS-9など足もとにも及びません。CやLISPなどの言語なども，マイコン上で走る言語はすべてCP/M-80 上で走ります。また，ソフトの質も8ビット CPU の中では最高のものが入手できます。もちろん，お値段も結構なものですが……。

ただ，今となっては時代遅れのOSになってしまったような気がします。逆にいえば，80系8ビット CPU ではこの程度かなとも思います。

▶FLEX

なんとなく薄幸のOSという感じがします。むしろ，モトローラの68系CPUが薄幸のCPUといえるかもしれません。どう見ても，80系よりも68系のCPUの方が使いやすい(もちろん，用途にもよりますが)と思うのですが，登場が数年遅れたために流通ソフトで大きく水をあけられてしまいました。しかしもしあなたが，6809の勉強をして機械語プログラムの開発をしようと思うならば，FLEXがベストだと思います。

▶OS-9

「遅すぎたOS」という感じがします。優れた設計思想とCPUの高い性能とで，他のOSとは同列で比較することができないほど，よくできています。レベル2を用いれば，8ビット CPU の壁である64Kバイトを楽々と破ってしまいます。ただ，「8ビットのCPUでここまでやる必要があるのか」とか「5インチディスクじゃ遅くてどうしようもない」とか言われると，返事のしようがありません。また，初めてOSを使うような人にとっては，ちょっと難しすぎるかもしれません。

ただ，FMシリーズのユーザならば，使ってみる値打ちは充分あると思います。16ビットや32ビット CPU のOSは，おそらくUNIX系OSになると思われます。現在，一般のユーザが入手できる唯一のUNIX系OSがOS-9です。UNIXに興味がある人

や，8ビット CPU の能力を100％引き出してみたいというような人には，おすすめ品だと思います。

▶UCSD p-System

Pascal言語の教育用に作られたOSですから，Pascal によるプログラム開発に関しては大変使いやすくできています。また，pコードにコンパイルされたプログラムは，p-System が走っているマシンならばCPUにかかわらず走らせることができます。標準装備されているPascalの言語仕様も高いレベルです。価格が他のOSに比べ高いのですが，Pascal コンパイラ付きであることを考慮すれば，決して高すぎるというほどではありません。日本ではあまり普及していないようですが，Pascal 言語を勉強してみたい人にはおすすめできます。

▶CP/M-86とMS-DOS

8086／8088 CPU が出始めたころは，CP/M-86しかなく，16ビットの世界でも，CP/Mがスタンダードになるかに見えましたが，IBM-PCに MS-DOS が採用されたあたりから形勢が変化してきたようです。高級言語やユーティリティなど，CP/M-86とMS-DOSでは，もはや量・質ともに差はありません。OS 自体を見ると，はっきりいってCP/M-86の方が見劣りします。ただ，本質的な部分は大差ないので，今後デジタル・リサーチ社がどこまで機能拡張するか，マイクロソフト社がどこまで逃げ切れるかが興味を引くところです。

おしまい

それぞれ長所や短所があり，どれを選んでよいのか難しいところです。OS 選びで悩むなどというのは，FMシリーズのユーザのみが味わえるぜいたくではないでしょうか。どれを選んでも，何か得したような損したような気分になるものです。ともかく，「エイヤッ」という感じで，どれか一つを買ってきて，使いこなすところから始めてみませんか。OS の世界に足を踏み入れることで，あなたのパソコンも，新しい広がりを持つようになります。

大学生のためのソフトバンク就職セミナー

「パーソナルコンピュータの現状と今後の展開」

パーソナルコンピュータの昨年国内出荷台数は約100万台。今年は150万台が見こまれ、累積300万台にも達しようとしています。ついては、これから社会人になる大学4年生を対象に、パソコン市場の展望、パソコンソフト流通のしくみ、パソコン出版社の動向などに関する就職セミナーを開催します。パソコン業界に関心のある諸君は、ご参加下さい。

日　時：9月10日(月)・17日(月)
PM 0:30〜4:00

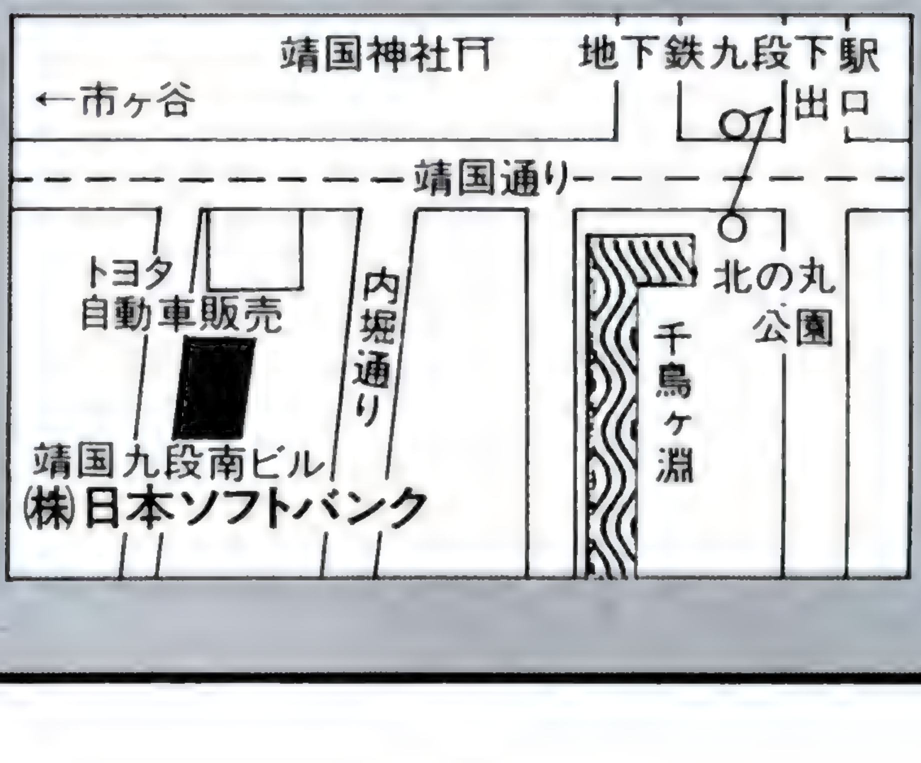
場　所：日本ソフトバンク本社会議室
千代田区九段南2-3-14 靖国九段南ビル
地下鉄九段下駅より徒歩7分、国鉄市ヶ谷駅より徒歩12分

対　象：Oh！シリーズ読者の大学4年生

参加料：無料

事前に電話にて連絡して下さい。

連絡先：日本ソフトバンク社長室／上橋
TEL 03(263)3665



座談会

私のOS論

AD2にはOS-9が，BSにはCP/M-86が標準実装，日本語機能を持つOSの発売，OS-9の異常人気などOSの周辺がぜんにぎやかになってきた。OSを無視してなにがハッカーか，というわけで，OSの達人を自認する9人に集まってもらい，それぞれのOSに対する思い入れなどを語ってもらった。以下はその座談会レポートである。なお，出席者および司会者の発言には独断と偏見が満ち満ちていることを最初にお断りしておく。

出席者
本郷　誠司
青山　千秋
志沢　雅彦
兼子　毅
阿部　久
吉川　厚
田中　明雄
平野　剛
(司会)
長沼　孝仁

司会（長沼）　今日は，OSを日頃使っているみなさんに集まっていただきました。OSについての感想やご意見を自由に出していただこうと思います。まず，みなさんが使ってきたOS,あるいは現在気にいっているOSについて，ひと言ずつ。

本郷　OSと言われても，ぼくは中型レベル以上はあまりいじったことがなくて，本格的なOSは知らないんです。マイコン上で動くOSでは，FLEXをいちばん最初に使い，次にCP/M，CP/M-86そしてOS-9ですね。どれも甲乙つけがたいのですが，気にいっているのはFLEXですね。いちばん最初のOSということで忘れがたい。

青山　大学で大型のOSを使ったことはありますが，非常に使いにくくて，最近はあまり使っていません。パソコンのOSで最近よく使っているのはMP/M-86，CP/Mのマルチユーザ・マルチタスクです。OS-9もちょぼちょぼ使い出したところです。MP/MのほうがCP/Mよりいろいろな点で便利ですね。

志沢　大型とかミニコンではUNIXを使ったことがありますが，本格的というほどではありません。マイコン上ではFLEXが最初で，最近はOS-9ですね。CP/MはWord Masterを使うときぐらい。いちばん使ったのはFLEXですが，今はOS-9にもっとも興味があります，評価は別として。

兼子　大学のVAX-11 UNIXを使う許可をもらったのですが，小文字が使える端末が手元にないので，まだ使えないでいるんです。マイコンのほうは，大学に入ってからFM-8を買ったので，使い出したのはFLEXよりOS-9のほうが先ですね。V1.1のときです。そのあと1年くらいたって，CP/M-80，CP/M-86を使う機会を得ました。いちばん使いやすいのは，最初に使い始めたということでOS-9です。

阿部　大型もミニコンも扱ったことはなく，もっぱらFM-7でやってます。いちばん最初に使ったのはFLEXなんですが，これはアセンブラを使うためで，OSを使うという目的ではないんです。エディタとアセンブラを使うためなんですね。OS-9も持っていますが，これもOSとして使ったわけではありません。ただ，OS-9には興味がありますね。OSはアプリケーションを動かすためのものだと考えると，OS-9にしろFLEXにしろ，それほどアプリケーションは出ていないので，本格的な評価が出るのはこれからなんではないでしょうか。

吉川　大型は一度も使ったことはありません。パソコンでCP/M-86を最初に使っていて，そのあとCP/M-80にソフトがいっぱいあるので使ってみた。次いでp-Systemを使い，1年前からMS-DOSを，最近はOS-9を2～3か月というところです。使いやすいという点ではp-Systemが気にいってます。

田中　パソコンでは3年ほど前に，仕事の関係で，使いたくなかったんだけどFLEXを使いました。68000のクロスアセンブラがFLEXにしかなかったので。もっぱら今はMS-DOSですね。去年はずっとCP/M-86を使ってきたんですが，これもゲームを作るためにしかたなくという感じ。OS-9は全然興味ないですから使ったことがありません。いちばん気に入っているのはやはりMS-DOSですね。

平野　大型もちょっとやってみたんですがすぐやめてしまって，このごろは，VAX上のUNIXと，PDP-11のRT-11というFLEXの元の形みたいな感じのモニタがあって，それを仕事に使っています。これはすっきりしてて使いやすい。マイコン関係ではFLEXとOS-9を使っています。ほかには，使いたくなかったんだけど，仕事でしかたなくCP/M-80を使ったことがあります(笑)。

OSはアプリケーションを超えるか

司会　お話を聞いてると，どうも好きでOSを使っている人はいないみたいですね。

本郷　正直いってOSの概念があまりよくわからない。逆にそういうのを感じさせないのがOSじゃないかとも思います。僕がFLEXを気に入っているというのは，OSとかDOSというイメージではなく，ただアセンブラとして気に入っているわけで。当時は富士通からアブソリュートアセンブラとか手頃なものが出ていませんでしたから。

司会　特定のアプリケーションソフトを使いたかった，ということですか。

本郷　そうですね。だから，FLEX上でBASICを走らせた，なんてことは一度もない。

司会　使いたいアプリケーションがあったので，OSを使わなければならなかったというわけですね。もっと積極的にOSを肯定される方はいらっしゃらない？

青山　OSは好きですよ。CP/M-86はいいですね。JWORDとF-BASIC86が付いていて。

本郷　付録に魅せられて買うという，なんか学研とかグリコとか…(笑)。でもそういうとこってありますね，絶対に。

司会　いちばん使ってみたいOSのアンケートで，けっこう人気が高かったのはOS-9だったんですね。 みなさんはCP/MみたいなOS――FLEXを含めて――と，OS-9みたいなOSとどっちが好きですか。

兼子　それは使い方によるんじゃないですか。何に使うかによって決まる。

本郷　誠司氏

司会　OS-9は何に使っているんですか。

志沢　DOSの解析！（笑）

阿部　その前にCP/M-86が好きだという人，MS-DOSが好きだという人，p-Systemが好きだという人とか，いろんな人がいるわけです。その好きな理由を知りたいですね。

田中　僕がMS-DOSが好きなのは，アプリケーションのほしいと思うのが何でもある。あとCP/M-86に比べると，DOSとしてもかなり洗練されているからです。そういう意味ではOS-9も同じでしょうが，アプリケーションがない。

青山　FM-11でのCP/M-86はいい。漢字はあるし，前のCP/M-80もZ-80カードを付ければ使えるし，CP/M-86は当然使えて，あとF-BASIC86を付ければF-BASICのファイルが読める。3種類に使えるというので非常にいい。

司会　p-Systemはどうですか。

吉川　第1に覚える必要がない。僕みたいに記憶力の容量の狭いのは全部覚えていられないんですよ。p-Systemだと全部画面に出てくるし，名前が全部出てくるから。名前がある程度わかっていればできますからね。画面に始めからHELP機能がついているようなもんだから楽です。あと標準でアセンブラとかすべてついてるし。そんなに欠点がない。確かに言語の面では少ないですが。

本郷　他の機種でも同じに扱える，という点では確かにp-Systemはいいですね。昔，BASICにも飽きてきて，ほかの高級言語が知りたいなっていうとき，たまたま友人がAPPLE IIを持っていて，僕もFM-8でp-Systemを手に入れていたんです。その頃はお互いに基本的なものしか持っていなかったので，まるっきり同じに扱えたというところが面白かった。たとえば，CP/MなどはI機種ごとに若干の違いがあるのは，しかたがないとは思うけれども，なんのためのOSかっていうことが出てくる。

長沼　孝仁氏

青山　言語の数の多さという点ではやはりCP/Mがナンバーワン。何でも使える。

兼子　何でも使えますか，ご自分で。

青山　たいてい使ってますよ。

兼子　それを使って仕事をですか。

青山　仕事はしない。勉強だけ。

司会　今の青山氏の発言にからめて，OSのアプリケーションの多さはよく言われますが，結局だれがそれを使っているかです。CP／Mもそうだけど，FORTRANが走るPascalが走ると言っても，いったい誰が使って，何に使うのかなあと思うのですが。

吉川　そのことで日立の人に嫌味言われましたよ。S1にFLEXとかOS-9とか載せるんですか，と聞いたら，誰かそんなの使える人がいるんですかって。どうせOSなんか初めから使えないという感じで。

司会　以前，CP／M-86上でFORTRANを使ったとき，文法はいいんですけど64Kのセグメントの問題で配列が自由に切れない，ということがあったんです。結局，みんな使えないからそれがバレないで存在してて，FORTRANがあるという幻想に結びついているのではないかと思いましたが。

兼子　FORTRANでプログラムを書かなければならない人が，マイコンユーザでどれだけいるか，ということではないです

青山 千秋氏

か。たとえば，会社にミニコンが入っていて，あるものしか走らないで，家で勉強するためにというならわかるけど。BASICもマスタしていないような人が，FORTRANだ，COBOLだ，Cだとかいって，CP/Mなら何でも走りますと言われたってね。大学生なら端末でFORTRAN使ってれば充分だし。家で使うならBASICで全然困らないだろうし。

田中 開発ツールだったら，いっぱいあったほうがいいんじゃないですか。

司会 開発ツールにしたって，要するにアセンブラでしょう。いいのが1つあればそれですむんじゃないですか。

田中 エディタ，アセンブラ，デバッガのいいのが揃っていればいいけど，ないのもあるでしょう。いっぱいあるということは，自分のほしいものもあるということですから，多いに越したことはなく，少なければいいというものでもない。

阿部 今，話を聞いていても，p-Systemは別としても，CPUに何を使うかによって違ってきますよね。たとえば6809で使うんだったらFLEXとOS-9しか今のところないわけで，そのどちらかということになる。80系を使うんであれば，ここにいるお2人とも，16ビットのCPUを開発するために使っている。そういうわけで，分けていかないと。どのCPUについてどうなのかと…。

司会 今回はCPUのことは置いといて，DOSを中心にいきましょう。

志沢 FMならカードを買えば何でも動くからあまり関係ないですね。ソフトのほうが，むしろ高いですよ。

司会 あらためてアプリケーションのことなんだけど，今まで使ってみて，これが良かったなというのがありますか。

青山 Word Starとか，CONDORもなかなか捨てがたい。

司会 データベースとかワープロみたいなものだと，けっこうアプリケーションは必要とされますが，アプリケーションで気に入ってるものはすべて言語以外のもの。アセンブラは言語だとは思えませんが。

吉川 CP/M-80で動くZAP-80というのがあります。ディスクトラックの1セクタとか一つひとつ読み書きもできるし，ファイルとか全部直しちゃうし，はじめからメッセージが出てくるんで，まったく覚える必要がない。

兼子 メニュ形式は，慣れないうちはいいけど，慣れてくるとうっとおしくなる。

吉川 わずらわしいけど，毎回，毎回忘れてますから。

司会 実際に使われるユーティリティというか，アプリケーションソフトは，どういうDOSが使いやすいのでしょうか。まず，エディタのほうから。

本郷 結局，慣れの問題だと思うんですけどね。それぞれに使いやすい。ある程度使えるように考えて作ってあるから，それぞれの長所もあるし，欠点もある。正直な話，僕がいちばん気に入っているのは，BASICが簡単でいいですね。DOSでといわれたらよく使ったFLEXなんかよかったと思う。ラインエディタなんか…。ワープロ的にも使えたし。

司会 僕はスクリーンエディタが好きですが。

志沢 RT-11のKEDというエディタはいいですね。キーボードにカバーがかぶせられてて。いい端末だから。スクロールもきれいだし。

平野 スクロールしないとだめですよね。BASICって，1回出すともう1度出し直さないと，いくらスクロールダウンさせようと思ってもできないでしょ。

青山 KEDでいいのは消したのが残っている…。

平野 あれをつけるのは簡単なのに，なんであんまりないんだろうね。

司会 OS-9のスクリーンエディタはどうですか。

兼子 あるけど使ったことがない。ヘルプモードがあって，コマンドは全部スクリーン上で確認できるみたいです。

司会 FLEXなんかどうなんですか。

本郷 スクリーンエディタはあるそうですけどね。

司会 ほぼみんな使ってないということは，普及してないということかな。

本郷 日本語FLEXはスクリーンエディタですね。

司会 日本語FLEXは20行で出ますか？

本郷 FM-7，FM-8用だけで，12.5行。

司会 あれは英字モードとかいって20にならないんですか？

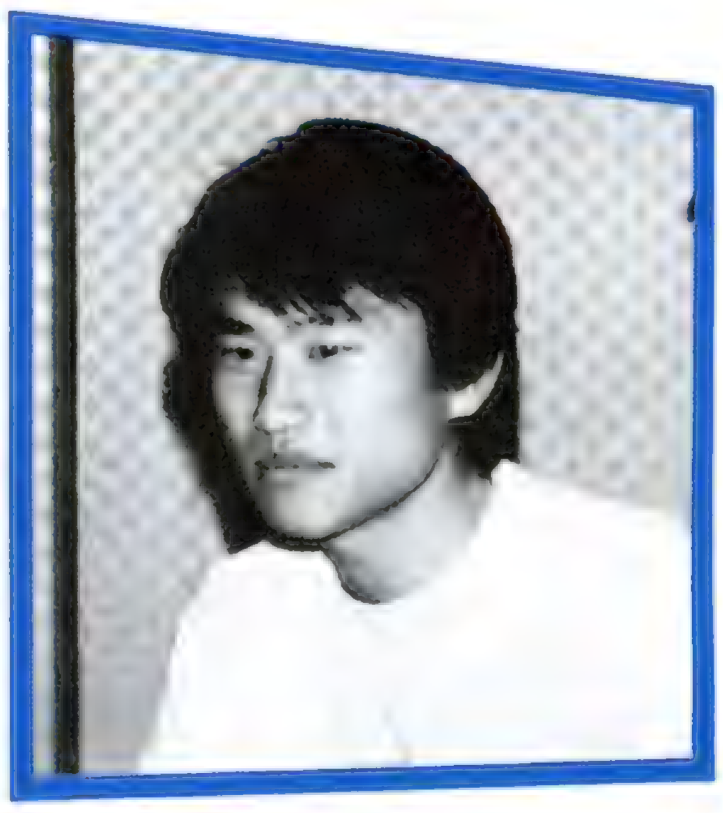
本郷 簡単に切り換わりますよ。だけど起動時にはアルファベットも16ドットのやつが出てくるので，見た瞬間びっくりする。

兼子 あれも1時間ぐらいじっと見てると慣れてくるから不思議なもんです。で，横40文字，縦12.5でずるずるスクロールしてくる。

司会 MS-DOSはどうですか？

田中 Word Masterですね，使っているのは。Word Starは，いろいろなファンクションがあってもスピードが遅いのでイライラしてきますが，その点，Word Masterは単純だけど速いんです。

司会 p-Systemのエディタはどうなんで



吉川 厚氏

しょう

吉川　そりゃあ，いいです。

司会　OSは機械を換えても一応動くように設計されているから，すごく遅い。

青山　Vエディットというエディタはサブシステムで1行だけあけるエスケープシーケンスを使っていて，一瞬にしてオープンラインできるから気分がいい。

司会　アセンブラなんかどうです。

兼子　OS-9はアセンブラで書くべきではない。

志沢　アセンブラでマルチユーザでやったりしたらよくないでしょうね。(笑)

兼子　あれは，BASIC 09がバンドリングされているということは，それで書きましょうということですよ(笑)。

田中明雄氏

マルチユーザ・マルチタスク・マルチウインドウ

司会　OS-9というOSが今，非常に話題を呼んでいますが，これがほんとうのマルチユーザなんですよね。だけど，その辺がほとんど理解できない。それ以上に何でマルチユーザなのか，マルチタスクなのかわからないんです。何でそういうものがみんな好きなんですかね。

兼子　打ち出すのをじっと待つのが嫌だから。プリンタだけでなく，ディスクにファイルを書きたいときに書いといて，その間に別の仕事ができるとか。

本郷　マルチユーザは大げさにしてもマルチタスクぐらいはね……許せる。

兼子　マルチタスクしてて，タスクにユーザ番号を2バイトつければいいんだから。マルチユーザとマルチタスクと別に違わないでしょう。マルチタスクできればマルチユーザになってるんで…。

司会　マルチタスク自体はあったほうが便利ですかね。

兼子　あると使う。なきゃないで別に…。

司会　あると使いますか。

兼子　使いますよ。せいぜい2つぐらいですが。レベル1だったら2つが限度かな。この間のマイコンショウで，OS-9上で走るインテグレーテッドソフトNOA110というのは，要するに，OS-9のマルチタスク機能を使ってプリンタで打ち出しながらでもエディット作業ができます，とやってたから。

司会　プリンタなんか遅い装置だから，それを裏でやれればいいんですよ。でもディスクだと，ディスクを使うジョブと使わないジョブが明確に分かれているというのはあまりない。そうなると結局，ディスク待ちになっちゃって，同時に進行できないこともあるでしょうね。プリンタに関しては確かにいいと思うけれど，スプーラがあれば充分じゃないかと思うんです。

兼子　レベル1ならマルチタスクとかは必要ないでしょうね。

阿部　スプーラができるようにすれば，OS-9でマルチタスクになるわけですよ。マルチタスクできれば，マルチユーザになる。だから，必要だから作ったというよりも，システムの性格だろうと思いますね。スプーラを効果的にやろうとしたら，マルチタスクをサポートするようにしたほうができるんじゃないだろうか。

司会　マルチタスクは裏方としてそれ以上は使わないという，あってもなくてもどうでもいいということですか。すると商品としてマルチタスクだからいいよというのは，疑問を感じますね。

兼子　シングルユーザで使っている限りはね。聞くところによると，FM-11に端末4台ぐらいつなげて，ハードディスクつなげてやってるところがあるそうですよ。

阿部　FMパソコン祭りで実際にやってましたよ。

司会　それによって得られるメリットがマイコンにあるんですか。

兼子　安い。(笑い)

阿部　システム開発じゃ，楽になるんではないですか。安くなるでしょう。だって，ファイルがいっぱいあるでしょう。それを共用できるわけですから。

司会　それによるメリットって大きいんですか。

阿部　大きいと思います。

志沢　OS-9の場合は，マルチウインドウと組み合わせたときに非常に使いやすい。たとえば，ひとつのプログラムを開発する場合でも，デバッグするのにすごく時間がかかるのがあるでしょう。それを別のウインドウで動かしておいて，あとはリストを見てとか，そういうことができる。

平野　それとかエディタを速くやろうと思ったら，ひとつのファイルの長さをだいたい2，3ページにおさえて，バラバラにしたほうがよい。そういうときエディタをいくつも走らせてもいいし。

司会　マルチウインドウというのは，2つまでは有利ですね。

志沢　3つあっても，普通は2つで使っていて，3つ目はほかのウインドウに重ねといて，非常の場合に使える。

兼子　ウインドウは重ねて使うでしょ。あれは待避用のを作っておくと暴走しても必ずシステムに戻れるから。レベル2はたぶん大丈夫だと思うけど。

兼子 毅氏

志沢　とはいえハードによるんじゃないですかね。ソフトウェアインタラプトがかかったときに，フラグが動けばいいんだけど…。

兼子　結局，スーパバイザモード，つまりシステムに行っちゃったときに暴走していると変なことが起こるから。

志沢　6809だと6829がついて，ちゃんとしたシステムだとメモリ空間が分けられるからできるんだけど。FM-11じゃできないのかな。

本郷　そういう設計のもとで，どうして29はできないの？

兼子　29って日本にあるの？　だってこの間のマイコンショウのモトローラのブースに29だけなかった。

司会　そういったハードウェアの制約も大きいですね。CPU 自体が，スーパバイザというものを考えているものもあるのに。

本郷　OS-9 が出たとき，6809の限界に挑戦してみようかなという感じで作ったのかなって思った。最初の設計思想に忠実に，それだけでできたのかなという気がありました。

兼子　だって，開発がそうでしょう。BASIC 09 をモトローラとマイクロウェアが作って，これがのるOS がないからって作ったのが OS-9 だから。

司会　OS-9 の特徴として階層ディレクトリというのがあるんですが。

本郷　OS-9の階層的ディレクトリというけれども，それをチェックするルーチンがあったら，残すほうの階層につながっていくとか，そういうのが見れたら視覚に訴えてわかりやすいような気がするんですけど。

司会　MS-DOS はついていますね。ツリー。でもいちばん上からしか見えない。

兼子　OS-9 のコマンドは使いこなせないんですよね。あれもできる，これもできると言っても，誰も使っていない。

司会　田中さんのようにハードディスクをお持ちの方はいいのですがね。

本郷　マイコンのハードディスクといったら，楽しみはあれを持ち歩いて見せびらかすことですから。(笑い)　持ち運びしやすいからと，あれだけ持ち運んでいる人がいますよ。

主観的OS論

司会　この辺でOS 全体を総括して，どのOS がいいか，主観的にお話しください。

本郷　FM ということを考えたら FLEX か OS-9 ぐらいを使わざるを得ないと思うんですよ。OS-9の場合は，最近，富士通から出ている7用とかけっこう安いですし。それに AD 2 なんか最初からついてきちゃいますし。だからこそ関心が集まっているのだと思います。安いし，それなりのものを持っているし，説明書もいっぱいついてくるし。そういうのはユーザから見たら何かすごそうだなという感じで興味の対象になるんじゃないでしょうか。僕は開発用に FLEX をよく使いました。FLEX はディスクがクラッシュしやすいという欠点がありますが，それだけ除けば FLEX は好きですね。11の FLEX はけっこう速いですし，その上で何かを走らせるということはなかったですが。

司会　OS-9 人気は異常だと僕なんか思うけど，やはり OS-9 がいちばんいいかどうか。

阿部　難しいですね。開発する立場からいえば，もう少しユーティリティがほしい。星光さんも頑張って作っているようですし，それが出てくれば OS-9 は使いやすいものになると思います。

兼子　マルチタスクとかマルチユーザを使わなくてもね。

青山　OS-9で今まで出てなかったけどメモリモジュールが強力なのね。サブルーチンを作っておけば，LOADすればどれからでも呼べるというのは強力だと思う。

阿部　それとパイプライン。コマンドにもパイプラインがついている。

兼子　パイプは強力だと思います，メモリさえあれば。さっき言ったモジュール作って，RUN で呼び出すよりも，要するに上から下にデータが流れてくるんだったら，パイプライン使って走らせたほうがよい。

阿部　「ソフトウェア作法」から出ているああいうやり方を本格的にやろうと思ったら，OS-9でなかったらできないだろうと思います。

兼子　確かに，I/Oのリダイレクトができるのは強力です。僕は今，卒論のためにコンピュータでシミュレーションをやっているんですが，普通，2MBとか3MBのデータ持ってきて解析する場合は大型使わざるを得ないけど，僕は乱数を1000個とか2000個発生させてデータをとっているので，マイコンレベルでできるのですね。すると，乱数を発生させ，それを書き出すプログラムを作って読み取ったやつを解析して，たとえば平均出すとかのプログラムを作ってプリントさせるとか，真中だけ入れ替えればデータを加工して出してくれるとか，それは確かに便利ですよね。そうすると，スクリーン1枚でプログラムが書ける。自分で何やっているかわかる。BASIC だから遅いですがね。

司会　そういうのはCP/Mのほうがよっぽど楽ですよね。

兼子　マシン語開発しようと思ってOSの導入を考え，OS-9を入れるとしたら気違いじみているけど。せいぜいBASIC程度の言語で書いて何か仕事をしたいというなら，確かに慣れるまで大変で，僕なんか最初，ディレクトリさえとれなくて，どこにセーブされたのかわからなくてということがありました。しかし慣れてしまえば小さい仕事にけっこう使えますね。

阿部　マシン語でもモジュール化できますよね。CP/MにしてもMS-DOSにしても一度作ったものをアペンドして，再アセンブルにしないとだめなわけです。それがメモリモジュール化できますから，どんどん呼べばできる。

兼子　リンクがないということ，それは画期的でしょうね。

青山　モジュールの1つを新しく直して，普通のだったらリンクし直さないといけないのに，あれはリンクしないでいい。もう言うことはないですね。

司会　ほかのCPUのDOSも考えてOS-9が使いやすいと思いますか。

兼子　16ビット使う必要ってどこにあるんですかね。速い？　メモリがいっぱいある？

司会　僕は16ビット好きですね。

兼子　16ビットはいいんだけど，僕は8086嫌いです。だってセグメント間違えると暴走するんだもの。まあ，慣れの問題ですけどね。

司会　DOSについては，OS-9のできはいいんですか。

志沢　比較にはならない。ものが違うし，あとから出たんだから良くなくちゃおかしい。(笑い)

兼子　僕の友達でずっとCP/M使ってたんだけど，OS-9を使わせてみたら，これはきれいすぎる，こんなのマイコンで走らせちゃいけないと言って，2日で返してきましたよ。

志沢　雅彦氏

司会　アプリケーションの多いDOSは当然あります。CP/Mは多いけど，OS-9は今のところ少ないですね。その点，不便はないですか。

兼子　今は不便ですね。どこまで富士通なり星光が力を入れるかですね。

OSの普及と可能性

司会　ところで，OS自体が普及するかどうか，今のところ疑問なんです。今はBASICで走るのが普通で，それとは別にOSがあるわけです。そういう形でもってOSというものが，BASICがバンドリングされている形とは別に普及するかどうか。それほどOSが必要だとは思わないし。BASICがのっているマシンとOSがのっているマシンを比較してどうですか。やっぱりOSは必要ですか。

本郷　僕がいちばん最初に言ったみたいに，意識させないでということでOS-9が人気を持っているのはBASIC09を積んでいたりとか，OSということを考えないでも，BASIC09でマルチウインドウができる，マルチタスクができるとか，そういう方面でみんなが見てるんじゃないかと思います。OSというイメージはないんじゃないですか。OS-9を，使ってみたいOSとは言ってますが，そういうふうにはとらえてなくて，ソフトという意味でしかとってないんじゃないかな。

平野　どういうふうに使えるかは考えていない。

本郷　ええ，でもそういうふうに意識させないOSとして広がっていってもいいのかなという気もしますね。

司会　OS自体はマイコン初心者にはなじまないと思うんですが。

本郷　そうだと思う。

司会　dBASE-IIみたいにいいソフトがあっても，それがOSにのってしまった以上，もうおしまいじゃないかという気がします。

本郷　dBASE-IIを走らせるためにしかたなくという考え方の人が多いんじゃないかな。それもBASIC09で何かやってみたいとするとOS-9を走らせなければならないというように，アプリケーション走らせるためにOSがあるという感じですね。

平野　アプリケーションだけだと，作表とワープロとデータベースが一緒になった統合化ソフトというやつがアメリカで売っている。そうなってくると，OSの意義はなくなっちゃうね。

兼子　OSがあると，ソフト屋さんがちょっとは楽できるでしょうね。

平野　そうすると，OS-9があるからといって，ソフト屋さんがOS-9上でやろうという動機はまるでないんだ，OS-9は主にFMシリーズでしか動いていないんだから。そうするといくらOSが出てきても普及はしない。

兼子　それはソフト屋さんの考えしだいでしょうね。けっこうOS-9も売れてきたから，OS-9上で走るソフトを作っても売れると考えるか。CP/M-86なんかで動くソフトでも，98だけの拡張サービスコードを使ってるとかあるでしょう。そうするとどうなんです。98だけでみればいいけど，CP/Mのソフトウェアバスとしての役割りは日本で

はなくなってきているでしょう。たとえば，生のままじゃCP/M-80で作ったファイルは読めない。ユーティリティが出ているということは，それは，ソフトウェアバスでなくなっていることでしょう。ユーティリティで使えるんだったら，OS-9でF-BASICを読めますよ。

司会 それぐらいの知識を持っていないと手が出せない。

兼子 DOSに手を出すんじゃなくて，DOSの上で走るソフトに手を出す。

司会 それではままならぬ。

吉川 だからやたらに手を出してはいけない。(笑)

兼子 残るのはMSX-DOSだけですか，まだ出てはいないけどね。

志沢 最後の救世主ですね。

兼子 ハードも決めちゃってるというすごいOS企画ですから。

司会 OSはソフトウェアバスとしての意味がなくなっているのに，やっぱりソフトウェアバスとしての中途半端な意味を持っている。だから，OSにのっていて，グラフィックを使うのってないんですよ。グラフィックはパソコンにとって重要な機能ですよ。

兼子 グラフィック使いたかったらスタンドアロンBASICがある。

阿部 グラフィックでもOS-9の場合はドライバが書きやすい。

兼子 もちろん自分の機械の上ではできるけど，ほかの機械の上では必ずしも走らないから。たとえば98のグラフィックにあわせたCP/M-86なりMS-DOSで走るソフトを作ってしまうと，どんなにOS上でグラフィックをサポートしてても，グラフィックがないMS-DOSマシンじゃ走らないということになる。

吉川 コンカレントCP/Mはグラフィックをサポートするという話です。

司会 OSは結局マニア以外の人にはうけないのに，何かそれを錯覚して使いたがっている人が多すぎると思うのです。OSが普及するかどうか，生き残れるかどうかは，そういうところにカギがあるような気がします。今は，開発ツールとしてのOSと，アプリケーションを使うためのOSとがごっちゃになっているように思いますが。

阿部 いろいろな論争があるのは，ほとんど開発者側ですからね。

司会 コンカレントになると，マルチウインドウをサポートしているOSが多いんだけど，使いやすくなるんですかね。

青山 たまに便利じゃないですか。

阿部 マイコンをゲームマシンとしてでなく使ってゆくためには，やはりOSがあったほうが開発側にとっても，ユーザにとってもいいとは思うんですよ。多くの機種で動けるようになれば安くなるとは思うんですよ。そりゃソフトウェアバスの問題はありますけどね。

青山 少なくともOSの思想があるパソコンのほうがいいといえると思います。IBM 5550の古いやつは，ワープロのファイルがBASICで読めない。だがCP/M-86ならばJ-WORDで作ったものがデータベースで読める。そういうことができるからOSの思想は必要だけど，ユーザには関係ない。

司会 開発者のためのOSと，アプリケーションユーザのためのOSと分けて考えてみても，開発者用のOSとして終わってしまうんですかね。この頃のOSの傾向は漢字を使ったりとか，ウインドウを用意してアピールしてるんだけど，どうですかね。

兼子 無駄。

平野 普通のユーザには関係ないもんね。

阿部 もともとOSというのはソフトウェア開発の負担を減らすためにあるわけですからね。

平野 そうなると素人立入り禁止ですか(笑い)。でも，素人にもいろいろある。機械がさわれる素人なら勉強すればわかります。OSだって，普通ディレクトリが見れて，リストがとれて，エディットができて，コンパイルできれば，それで使えないことはない。それとあとワープロが使えればね。

平野 剛氏

ユーティリティがよければ素人だって使えるでしょう。

兼子 別に，素人には使えないから止めたほうがよいと心配してあげる必要はないんじゃないですか。興味があるんだったら，OS-9を買って，毎日リセットボタンを押して半年ぐらいやってれば，プログラムの1つも書けるようになるだろうし。マルチタスクで自分のプログラムができれば，それはそれでいいんだから，誰が何といおうとやればよい。

阿部 使ってみたい気持ちがあって小遣いのある人は大いに楽しんだらいいと思いますね。だけど，使いこなすだけの努力は覚悟してほしい。

志沢 興味があるなら使ってみて，だめだったらBASICに戻ればいいじゃないですか。

青山 でも，BASICしかわからないという事態は避けてほしいと思いますが。

田中 これからのOSは，一部のマニアが俺は使っているんだとニタニタ笑いながら使っていくことになるでしょう。

平野 OSを使う人のタイプがあるみたい。思考型の，まずマニュアルから読み始める人はまずだめで，まず，さわってみて進めていく直感型の人はうまくなる可能性があるようです。

司会 今日はどうもありがとうございました。

PC-9801の5インチディスクをFMで読む for CP/M-86

紀尾井　太郎

ディスクファイルの互換性

CP/Mはソフトウェアバスと言われるように，機種に依存しないで動くプログラムを書くことのできるOSです。しかし，プログラムを記録している媒体であるフロッピディスクは，8インチの片面単密度のもの以外ではフォーマットが統一されていません。8インチのディスクがあれば，片面単密度のディスケットにファイルを移すことにより他の機種でもそのファイルを読み書きできるため，それほど不便ではありません。しかし，5インチのディスクしか持っていない人にとっては，同じCP/Mマシンといえどもファイルを読みとれなくて不便なものです。

PC-9801とFM-8,11は，どちらもCP/M-86が動きます。両機種のあいだでは，8インチフロッピなら両面倍密度でも互換性がありますが，5インチ2Dフロッピでは互換性がありません。そこで，CP/M-86におけるディスクのアクセス方法を説明し，PCで5インチ2Dフロッピディスクに書いたファイルをFMで読み書きできるようにするプログラムを紹介します。

ディスクパラメータテーブルの書き換え

CP/Mはディスクに関する2つのテーブルを，BIOS内に持っています（図1参照）。

図1　ディスクパラメータテーブル

(1)ディスクパラメータヘッダ

XLT	セクタ変換テーブルのアドレスが入る。ディスクアクセスの効率を上げるために順番にセクタを置かないことがある。
DIRBUF, CSV,ALV	BDOSで用いる作業領域のアドレスが入る。
DPB	ディスクパラメータブロックのアドレスが入る。

(2)ディスクパラメータブロック

SPT	1トラック当たりのセクタ数
BSH, BLM	何セクタを1まとまりと考えるかによって決まる。ディスクを管理するときはFLEXと違い，F-BASICのクラスタのようにいくつかのセクタをまとめて扱っていて，その値をデータブロックアロケーションサイズという。
EXM	ディスクの記憶容量とデータブロックアロケーションサイズによって決まる。
DSM	ディスクの記憶容量が入る。
DRM	ディスクに作れるファイルの数を入れる。F-BASICのディレクトリにはファイルのある先頭のクラスタだけが入っているが，CP/Mではディレクトリにファイルのあるデータブロックの値がその数だけ入っている。そのためファイルの大きさによってディレクトリの大きさも変わってくるので，DRMの値だけファイルが作れるわけではない。
AL0, AL1	DRMとデータブロックによって決まる。
CKS	DRMによって決まる。ディスクを交換したことを知るためにディレクトリのチェックサムをBDOSは持っている。交換することのないハードディスクやキャッシュディスクにはチェックサムは不要なので，ここは0にしておく。
OFF	ディスクの外側から何トラックを使わないかを入れる。IPLプログラムを入れておくディスク上の場所がこれによって確保される。

FMとPCのあいだの違いは，ディスクパラメータブロックの値が違うために生じています。ディスクパラメータテーブルがディスクの使用方法を決めています。ですからこのテーブルの値を変えれば，両機種間に互換性を持たせることができます。

ディスクパラメータブロックの値をみると，**図2**のようになっています（ディスクパラメータテーブルの見方は，**「ディスクパラメータテーブルの調べ方」**を参照してください)。たとえば，SPTが0040と0020というように異なっています。しかしFMのSPTを単に0020にしたのでは，よくありません。というのはPCではディスクの表裏で2トラックとしているのですが，FMでは表裏で1トラックとなっているからです。OFFが0002というように同じになっていても，FMではディスクの外側に2トラック,PCでは1トラック確保されていて，実際の確保済みトラックの数は違っています。そこでFMのOFFを0001とし，SPTを0040とすることでディレクトリが読みとれるようになります。もう1つの違いはDSMで，これは単に009Bと書き直すだけです。

結局，**図3**のようにディスクパラメータブロックを直します。

シングルドライブにおいてはこれでもかまいませんが，デュアルドライブの場合，一方はFM用，もう一方はPC用とするためにはこれでは不充分です。こ

図2　FM, PCのディスクパラメータブロックの値

	SPT		BSH	BLM	EXM	DSM		DRM		ALO	ALI	CKS		OFF	
FM：	40	00	04	0F	01	97	00	7F	00	C0	00	20	00	02	00
PC：	20	00	04	0F	01	9B	00	7F	00	C0	00	20	00	02	00

図3　ディスクパラメータブロックの修正値(FM用)

FM：40　00　0F　0F　01　9B　00　7F　00　C0　00　20　00　01　00

のままでは，２つとも FM用か PC用になってしまいます。ディスクパラメータヘッダのDPBは，同じディスク特性をもつドライブについて同じディスクパラメータブロックのアドレスが入っているからです。つまり複数のドライブが共通のテーブルを使っているわけで，そのテーブルを書き換えると関係ないドライブまで影響を受けてしまいます。ですから別々のファイルタイプのフロッピを同時に使うためには，もう１つ別のディスクパラメータブロックを用意する必要があります。FMのBIOSを見ると $40 あたりの番地は使われていないようなので，ここにディスクパラメータブロックを書くことにし，もとのディスクパラメータブロックは書き変えずに，DPBが新しいディスクパラメータブロックを指すように書き換えることにします。こうすることで，２台のディスクを FM用と PC用とに使い分けることができるようになります。以上のことをするのが，**リスト１**に示したプログラムです。

おわりに

PCとFMの間では確保済みトラックの数だけが違っていたのでディスクパラメータブロックだけ書き換えることで済みましたが，MULTI-16のファイルを読むようにするにはセクタ変換テーブルを作る必要があります。しかし１セクタが 256 バイトのディスクならこのプログラムを修正すれば，たいてい使えるようになるはずです。なお，このプログラムを PC でそのまま使っても動きません。OFFとSPT を PC用にすれば動きます。PC の方が８Kバイトだけ容量が大きいので，普段は PC のディスクフォーマットで使う方がいいかもしれません。なお FM用に買ったソフトは，FMだけで使うようにしてください。

▶ディスクパラメータテーブルの調べ方

まず DDT86を起動します（下線部を入力してください）。

```
A>DDT86↵
DDT86 1.1
```

アセンブルコマンドを使ってBDOSコールをします。機能コードの50番を使って，BIOSを呼び出します。

```
-A1000：0↵
1000：0000 MOV CL,32↵   ←機能コードをレジスタCLにセット。32は16進数。
1000：0002 MOV DX,100↵  ←BIOSコールパラメータのアドレスをレジスタDXにセット。
1000：0005 INT EO↵      ←BDOSを呼び出す。
1000：0007 ↵
```

セットコマンドを使ってBIOSコールパラメータを設定し，機能番号９のSELDSK を呼び出します。CX レジスタにはディスクのドライブ番号を入れます。ドライブAなら0を，Dなら３を入れます。

```
-S1000：100 ↵
1000：0100 00 09↵
1000：0101 00 01↵   ←この場合はドライブBを指定している。
1000：0101 00 ↵
```

イグザミンコマンドを使ってレジスタを設定します。1000：0 からアセンブルしたのでCSレジスタには1000を，IPレジスタには0を入れます。BIOSコールパラメータを1000：100に入れたので，DSレジスタに1000を入れます。

```
-XCS↵
CS 0000 1000↵
DS 0000 1000↵
SS 0000 ↵
ES 0000 ↵
IP 0000 ↵
```

トレースコマンドを使って1命令を実行してみます。

```
-T ↵
          AX   BX   CX   DX   SP   BP   SI   DI   IP
-I ------0000 0000 0000 0000 092C 0000 0000 0000 0000  MOV  CL, 32
*1000：0002
```

正しく実行されているようなので，ブレークポイントを指定してゴーコマンドを実行します。

```
-G,7↵
*1000：0007
```

イグザミンコマンドでレジスタを表示させてみます。

```
-X↵
        AX   BX   CX   DX   SP   BP   SI   DI   CS   DS   SS   ES   IP
-I ----0000 635B 0004 0000 092C 0000 0000 0000 1000 1000 0050 0050 0007
ADD  [BX+SI], AL
```

BXレジスタにはディスクパラメータヘッダのアドレスが入っています。ESレジスタが50ですから50：635Bにディスクパラメータヘッダがあります。

```
-D50：635B, 635A↵
0050：635B 00 00 00 00 00 00 00 00 8E 67 4B 64 9D AA BD AA.........gkd......
```

DPBが644Bですからディスクパラメータブロックは50：644Bにあります。

```
-D644B, 644A↵
0050：644B 40 00 04 0F 01 97 00 7F 00 C0 00 20 00 02 00 68 @..............h
```

これがディスクパラメータブロックです。

最後にコントロールCを打ってシステムに戻ります。

```
-^C
A>
```

リスト1

©1984 T.KIOI

```
CP/M ASM86 1.1   SOURCE: CHDPT.A86

                              ;
                              ;       ディスクパラメータテーブルをFM-11用とPC-9801用
                              ;       に書き換える
                              ;
                              ;

                                      cseg

                              ;       ドライブNo.を設定する

0000 8A0E8000                         mov     cl,cmdln
0004 BE8200                           mov     si,offset cmdln+2
0007 E8A600         00B0              call    getch
000A 3C20                             cmp     al,' '
000C 7503           0011              jne     skip0
000E E9C200         00D3              jmp     errcmd
                              skip0:
0011 B400                             mov     ah,0
0013 A27401                           mov     drive,al
0016 2C41                             sub     al,'A'
0018 A3B401                           mov     cxreg,ax

                              ;       FM-11かPC-9801かを設定する

001B E89200         00B0              call    getch
001E A2BA01                           mov     machine,al
0021 3C50                             cmp     al,'P'
0023 7407           002C              je      setdpb
0025 3C46                             cmp     al,'F'
0027 7403           002C              je      setdpb
0029 E9A700         00D3              jmp     errcmd


                              ;       ディスクパラメータブロックをBIOS内に書く

                              setdpb:
002C B131                             mov     cl,49
002E CDE0                             int     224
0030 B90F00                           mov     cx,15
0033 BEA401                           mov     si,offset source
0036 BF4200                           mov     di,offset newdpb
0039 F3A4                             rep movs al,al

                              ;       指定したディスクが2Dミニフロッピディスクか調べる

003B B132                             mov     cl,50
003D BAB301                           mov     dx,offset func
0040 CDE0                             int     224
0042 891EB801                         mov     dpt,bx

0046 268B7F0A                         mov     di,es:0ah[bx]
004A 268B4505                         mov     ax,es:5[di]
004E 3D9700                           cmp     ax,0097h
0051 7408           005B              je      mini
0053 3D9B00                           cmp     ax,009bh
0056 7403           005B              je      mini
0058 E98100         00DC              jmp     errmini
                              mini:

                              ;       ディスクをリセットする

005B BA0100                           mov     dx,1
005E 8B0EB401                         mov     cx,cxreg
0062 D3E2                             shl     dx,cl
0064 B125                             mov     cl,37
0066 CDE0                             int     224

                              ;       ディスクパラメータテーブルにあるディスクパラメー
                              ;       タブロックへのポインタを設定する

0068 8B1EB801                         mov     bx,dpt
006C 803EBA0146                       cmp     machine,'F'
0071 7421           0094              je      fm

0073 BA6C01                           mov     dx,offset donemsg
0076 E86C00         00E5              call    sendmsg
0079 BA8D01                           mov     dx,offset donepc
007C E86600         00E5              call    sendmsg

007F 268B470A                         mov     ax,es:0ah[bx]
0083 3D4200                           cmp     ax,offset newdpb
0086 7404           008C              je      same
0088 26A34000                         mov     es:dpb,ax
                              same:
008C 26C7470A4200                     mov     es:word ptr 0ah[bx],offset newdpb
0092 EB18           00AC              jmps    exit

                              fm:
0094 BA6C01                           mov     dx,offset donemsg
0097 E84B00         00E5              call    sendmsg
009A BA7601                           mov     dx,offset donefm
009D E84500         00E5              call    sendmsg

00A0 26A14000                         mov     ax,es:dpb
00A4 23C0                             and     ax,ax
00A6 7404           00AC              jz      exit
00A8 2689470A                         mov     es:word ptr 0ah[bx],ax

                              exit:
00AC B100                             mov     cl,0
00AE CDE0                             int     224

                              ;       コマンド行から1文字持ってくる

                              getch:
00B0 80F900                           cmp     cl,0
00B3 741B           00D0              je      eof
00B5 8A04                             mov     al,[si]
00B7 46                               inc     si
00B8 FEC9                             dec     cl
00BA 3C20                             cmp     al,' '
00BC 74F2           00B0              je      getch

                              progress:
00BE 80F900                           cmp     cl,0
00C1 740C           00CF              je      exitgetch
00C3 8A24                             mov     ah,[si]
00C5 80FC20                           cmp     ah,' '
00C8 7405           00CF              je      exitgetch
00CA 46                               inc     si
00CB FEC9                             dec     cl
00CD EBEF           00BE              jmps    progress

                              exitgetch:
00CF C3                               ret

                              eof:
00D0 B020                             mov     al,' '
00D2 C3                               ret

                              errcmd:
00D3 BA0001                           mov     dx,offset errcmds
00D6 E80C00         00E5              call    sendmsg
00D9 E9D0FF         00AC              jmp     exit

                              errmini:
00DC BA3301                           mov     dx,offset errminis
00DF E80300         00E5              call    sendmsg
00E2 E9C7FF         00AC              jmp     exit

                              sendmsg:
00E5 50                               push    ax
00E6 53                               push    bx
00E7 51                               push    cx
00E8 B109                             mov     cl,9
00EA CDE0                             int     224
00EC 59                               pop     cx
00ED 5B                               pop     bx
00EE 58                               pop     ax
00EF C3                               ret

                                      dseg

                                      org     80h
0080                          cmdln   rb      80h

                                      org     100h
0100 8E67977095FB             errcmds         db   '使用方法：CHDPP␣␣ドライブ名␣␣
     9640203A2043                                   FMまたはPC',0dh,0ah,'$'
     484450542020
     202083688389
     8343837596BC
     202020202020
     464D2082DC82
     BD82CD205043
     0D0A24
0133 8E7792E882B5             errminis        db   '指定したドライブはミニフロッピディ
     82BD83688389                                   スクではありません',0dh,0ah,'$'
     8343837582CD
     837E836A8374
     838D83628373
     815B83668342
     8358834E82C5
     82CD82A082E8
     82DC82B982F1
     0D0A24
016C 836883898343             donemsg         db      'ドライブ'
     8375
0174                          drive           rb      1
0175 24                                       db      '$'
0176 82CD8265826C             donefm          db      'はFM用になりました',0dh,0ah,'$'
     977082C982C8
     82E882DC82B5
     82BD0D0A24
018D 82CD826F8262             donepc          db      'はPC用になりました',0dh,0ah,'$'
     977082C982C8
     82E882DC82B5
     82BD0D0A24
01A4 4000040F019B             source  db      40h,00h,04h,0fh,01h,9bh,00h,7fh,00h
     007F00
01AD C00020000100                     db      0c0h,00h,20h,00h,01h,00h

01B3 09                       func    db      9
01B4                          cxreg   rw      1
01B6                          dxreg   rw      1
01B8                          dpt     rw      1
01BA                          machine rb      1

                                      org     40h

0040                          dpb     rw      1
0042                          newdpb  rb      15

                                      end
```

OS-9サービスリクエストの使い方

for OS-9

青山　千秋

はじめに

OS-9は，BASICとマシン語のつながりがF-BASICよりも自然です。F-BASIC上でマシン語を使うときは，BASICでマシン語をロードしてこなくてはいけないのですが，OS-9では，OSがマシン語をディスクから呼び出してくれます。BASIC09には線を引く命令はもともとありませんが，F-BASICにあってBASIC09にない命令は，RUN　LINEとすることで線が引けるように自分で作れます。6809機械語マスタ'84でマシン語が使えるようになったら，OS-9でも，さっそくマシン語を使ってみましょう。またOS-9は持っているけど，F-BASIC上で動くアセンブラを持っていないという人も，BASIC09だけを使っていないで，マシン語でプログラムを作ってみてはどうでしょうか。

サービスリクエストの実際

それではまずマシン語の簡単なプログラムを**リスト1**に紹介します。これは標準入力ファイルからの入力を標準出力ファイルへ出力するものです。

いったいこれが何の役に立つのかというと，これだけのプログラムで，ファイルがコピーできます。OS-9には入出力のリダイレクトという機能があるので，単に標準入力パスから読んだデータを，標準出力パスへ書くだけでファイルのコピーができてしまいます。F-BASICでは，INPUTやPRINTはキーボードとCRTにしか，それぞれ使えません。BASIC09では，リダイレクトの機能により，実行時に変更できます。ところがこれを使ってファイルをCRTに表示させてみると，ラインフィードを行いません。BASIC09ではキャリッジリターンしか行の終わりにないからです。そこでキャリッジリターンが現れたらラインフィードも出力させるように直してみたのが**リスト2**です。これでファイルがCRTに表示できるようになりました。パスの指定方法が違いますが，OS-9のコマンドのCOPYとLISTと同じことをしています。

リスト1をもとにしてOS-9でのマシン語の書き方について説明しましょう。

OS-9でマシン語プログラムを作るためには4つの規則にしたがわなくてはいけません。

1．絶対番地（エクステンドアドレッシング，JMP，JSR）を使わない。
2．データ領域はOS-9から渡された所を使う。
3．メモリモジュールにする。

リスト1

©1984 C. AOYAMA

```
Microware OS-9 Assembler 2.1    00/00/00 00:03:38                    Page 001
pip - OS-9 System Symbol Definitions

00001                                  nam    pip
00002                                  ifp1
00004                                  endc
00005     0000 87CD002F                mod    msize,name,type,revs,start,size
00006     000D 7069F0        name      fcs    /pip/
00007     0011               type      set    prgrm+objct
00008     0081               revs      set    reent+1
00009 D 0000                 buffer    rmb    1
00010 D 0001                           rmb    250
00011 D 00FB                 size      equ    .
00012     0010               start     equ    *
00013     0010 30C4                    leax   buffer,u
00014     0012 108E0001                ldy    #1
00015     0016 8600                    lda    #0
00016     0018 103F89        nextc     os9    i$read
00017     001B 2507                    bcs    exit
00018     001D 4C                      inca
00019     001E 103F8A                  os9    i$write
00020     0021 4A                      deca
00021     0022 24F4                    bcc    nextc
00022     0024 C1D3          exit      cmpb   #e$eof
00023     0026 2601                    bne    error
00024     0028 5F                      clrb
00025     0029 103F06        error     os9    f$exit
00026     002C 975252                  emod
00027     002F               msize     equ    *

00000 error(s)
00000 warning(s)
$002F 00047 program bytes generated
$00FB 00251 data bytes allocated
$23B6 09142 bytes used for symbols
```

4．入出力にはOS-9のサービスリクエストを使う。

OS-9ではプログラムがメモリ上のどこにロードされるか決まっていないので，JMPやJSRを使えませんし,6809では使う必要もありません。同様にデータ領域もわからないので,エクステンドアドレッシングも使えないのです。

OS-9はメモリ上にある1つのプログラムを，複数の人が同時に使えるようにしたので，プログラム領域とデータ領域が分離されています。ですから，プログラム領域にデータを書き込むようなプログラムはマルチユーザ，マルチタスクでは使えません。ただし，プログラム領域に定数を書くことは許されます。

メモリモジュールというのは，OS-9が管理するプログラムの単位でディスク上のファイルと同じようなものです。メモリモジュールとするためには，モジュールヘッダとCRCをプログラムにつけるのですが，アセンブラが，これらをつけてくれます。

入出力は直接I/Oをいじらずに，サービスリクエストを使います。CP/MのBDOSコールと同じようにして，サービスリクエストを呼び出します。I/Oに関する操作を直接行うときには，ドライバルーチンを書いて，キャラクタファイルマネージャに登録してから，そのドライバを通じて行う必要があります。

では**リスト3**を見てください。**リスト3**は，**リスト1**のソースプログラムです。

“nam”と“ttl”命令によってヘッダに表示する文字を変えます。左上に“pip” と表示されているのは，この命令を使ったからです。

“ifp1” は条件付きアセンブル命令で，“endc” までの行はパス1のときだけ有効，パス2では無効です。“ifp1” の他にも“ifeq”，“ifne”，“iflt” などがあります。“else” も使用できます。アセンブラはプログラムを2回読んでマシン語に翻訳しますが，1回目をパス1，2回目をパス2といいます。パス1ではシンボルテーブル（ラベルの値を記憶しているテーブル）を作り，パス2でマシン語に翻訳したものを出力します。パス2では不要となる部分を，“ifp1”と“endc” ではさむことにより，パス2で読みとばすので，アセンブル時間が短縮されます。

リスト1では表れていませんが,“ifp1” と“endc”の間には,“use”命令が使われています。“use/d0/defs/defsfile”とすることで，ここに“defsfile”が挿入されます。“defsfile” はシステムのシンボル定義をしているのです。このファイルはCの“stdio.h”のようなもので,CP/MでBDOSコールをするときに，ファンクションNo.を名前で書ける

リスト2

```
Microware OS-9 Assembler 2.1    00/00/00 00:05:00                    Page 001
crlf - OS-9 System Symbol Definitions

00001                                  nam    crlf
00002                                  ifp1
00004                                  endc
00005    0000 87CD0043                 mod    msize,name,type,revs,start,size
00006    000D 63726CE6     name        fcs    /crlf/
00007    0011              type        set    prgrm+objct
00008    0081              revs        set    reent+1
00009 D  0000              buffer      rmb    1
00010 D  0001                          rmb    250
00011 D  00FB              size        equ    .
00012    0011              start       equ    *
00013    0011 30C4                     leax   buffer,u
00014    0013 108E0001                 ldy    #1
00015    0017 8600         nextc       lda    #0
00016    0019 103F89                   os9    i$read
00017    001C 2512                     bcs    exit
00018    001E 8D18                     bsr    write
00019    0020 9600                     lda    buffer
00020    0022 810D                     cmpa   #$0d
00021    0024 26F1                     bne    nextc
00022    0026 860A                     lda    #$0a
00023    0028 9700                     sta    buffer
00024    002A 8D0C                     bsr    write
00025    002C 21E9                     brn    nextc
00026    002E 24E7                     bcc    nextc
00027    0030 C1D3         exit        cmpb   #e$eof
00028    0032 2601                     bne    error
00029    0034 5F                       clrb
00030    0035 103F06       error       os9    f$exit
00031    0038 8601         write       lda    #1
00032    003A 103F8A                   os9    i$write
00033    003D 25F1                     bcs    exit
00034    003F 39                       rts
00035    0040 DBC35F                   emod
00036    0043              msize       equ    *

00000 error(s)
00000 warning(s)
$0043 00067 program bytes generated
$00FB 00251 data bytes allocated
$23C5 09157 bytes used for symbols
```

ようにするため，あらかじめ“equ”によってNo.を定義しておくような部分がまとまっているファイルです。サービスリクエストをニモニック名で使うためには，このファイルが必要です。“mod”によってモジュールヘッダが作成されます。“mod”のあとには，6つの値が必要です。

1．モジュールの大きさ
2．モジュール名までの相対アドレス
3．モジュールの型と使用言語
4．モジュールの属性と改訂数
5．モジュールの開始アドレス
6．モジュールが必要とする最小の作業領域の大きさ

それぞれ“msize”,“name”,“type”,“revs”,“start”,“size”というラベルで指定しています。

“msize”は“emod”命令の直後に置きます。“msize equ *”によってモジュールの大きさがわかります。“*”は，プログラムアドレスカウンタの値を示しています。OS-9には“.”によって示されるデータアドレスカウンタもあります。プログラムアドレスカウンタは6809の命令を発生するたびに進められ，データアドレスカウンタは“rmb”などの命令によって，データ領域がとられるたびに進みます。たとえば，“ldy #1”という命令によってプログラムアドレスカウンタは2だけ増えます。そのためプログラムの最後に“msize equ *”と書くことで，モジュールの大きさがわかるわけです。

“name fcs /pip/”で，モジュールの名前が定義されます。“pip”という文字列をモジュールに書き込んで，プログラムアドレスカウンタを進めます。“fcs”によって“pip”という文字列の最後の文字の“p”の8ビット目が1となっています。

“type set prgrm+objct”で，このモジュールがプログラムモジュールで6809のオブジェクトコードであることを定義します。“prgrm”,“objct”は“defsflle”のなかで（正確には“OS-9 defs”の中で）定義されています。

また，“revs set reent+1”で，リエントラントで改訂数が1であるとしています。“reent”も“defsfile”で，定義されてます。これをつけなければ，リエントラントでないプログラムであることを示しますから，プログラム領域を書き直してもかまわないことになりますが，マルチタスクでは使えなくなります。改訂数を増やすことで，同一のモジュールがすでにメモリ上にあっても，古いモジュールを消さずに，使用できます。

前口上が終わって，データの定義に移ってきました。

“buffer rmb 1”で，1バイトの領域が確保されました。インフォメーションフィールドに“D”がついています（“00009 D 0000”の“D”）。この領域はデータの入出力時に使うものです。

“rmb 250”で，スタックのための領域を確保しておきます。サブルーチンとして使われるモジュールが，スタック上に変数領域をとりますから，これぐらいは確保する必要があります。

“size equ .”で，モジュールの必要とする最小データ領域が指定されます。

さて，プログラムの説明に入りましょう。

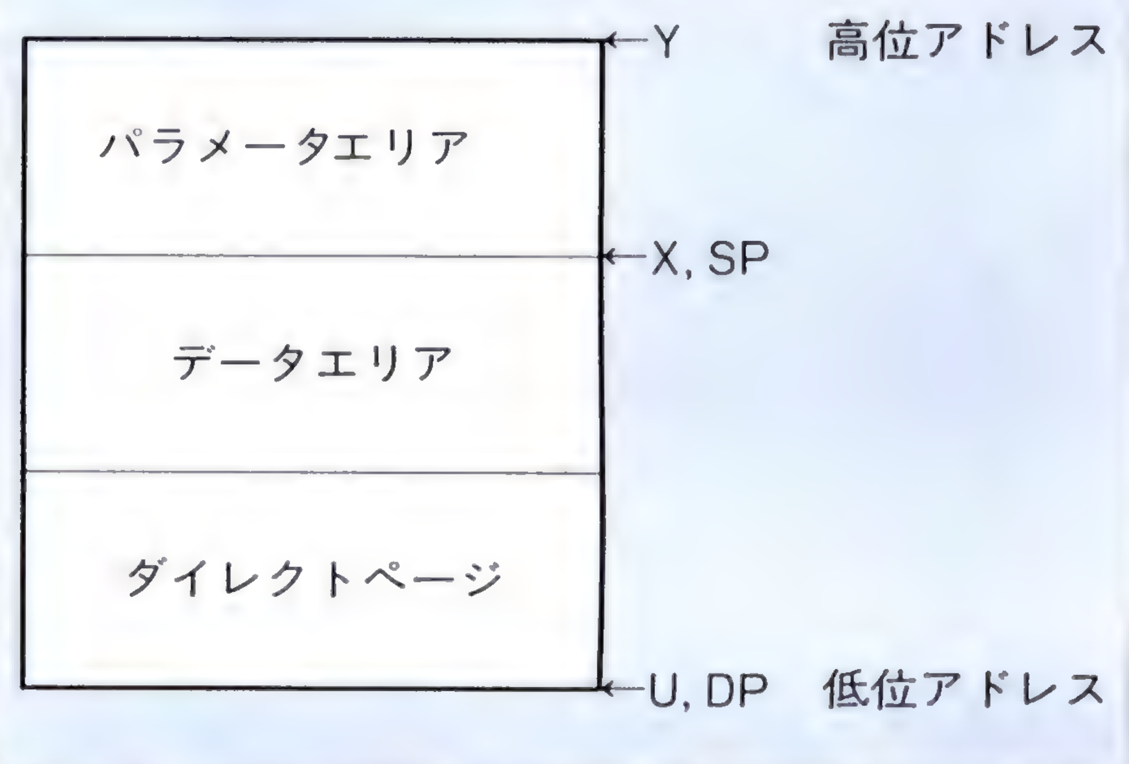
“start equ *”と“mod”によりプログラムはここから実行されます。

shellから制御が移されるとレジスタは図1のように設定されています。ダイレクトアドレッシングを使えばデータ領域の配置について意識しないで済みますが，ダイレクトアドレッシングを使わない場合は，インデックストアドレッシングを使わなくてはいけません。

“leax buffer, u”によってバッファの実アドレスがXレジスタに求まります。こうすることによりデータ領域がメモリ上のどこに割りあてられても，プログラムを修正しないで利用できるわけです。Z80などではこのようなことができないので，バンク切り換えを使って，マルチタスクを実現しています。

“os9 i$read”で，Yレジスタの値だけ，Aレジスタで示されるパス番号のパスより，データを読み込みます。そして，Xレジスタで示されるところに，書き込みます。パス番号は，“os9 i$open”を使って得られますが，標準入力パスは0で標準出力パスは1になっていて，ファイルを改め

図1 レジスタの設定



リスト3

```
  nam pip
  ifp1
  use /d0/defs/defsfile
  endc
  mod msize,name,type,revs,start,size
name fcs /pip/
type set prgrm+objct
revs set reent+1
buffer rmb 1
  rmb 250
size equ .
start equ *
  leax buffer,u
  ldy #1
  lda #0
nextc os9 i$read
  bcs exit
  inca
  os9 i$write
  deca
  bcc nextc
exit cmpb #e$eof
  bne error
  clrb
error os9 f$exit
  emod
msize equ *
```

てオープンしなくても使えます。

リスト1では“ldy #1”と“lda #0”により，標準入力パスから1文字だけ入力するようにしています。

エンドオブファイルになるか，エラーが生じるとキャリービットが1になるので，“bcs exit”によりループが終了します。

“inca”で出力パス番号を1にして，次の“os 9 i$write”で，標準出力パスへ出力するようにしています。

“bcc nextc”でエラーがないかぎり，次の文字を読みにいきます。そして“os 9 i$write”でエラーが発生するか，“bcs exit”により“cmpb #e$eof”に，制御が移ってきます。Bレジスタにはエラーコードが入っているので，エンドオブファイルが発生したかどうか調べており，“e$eof”は“defsfile”で定義されているのです。

もしもエンドオブファイルがおきていたら，エラーとしないで終了するために，Bレジスタを0にし，そうでない場合はBレジスタはそのままにしておきます。

“os9 f$exit”で，このプログラムは終了して，制御をこれを呼び出したプロセスに返します。たいがいの場合，shellに戻ります。Bレジスタにはステータスコードを入れて，この命令を呼び出します。0ならばエラーがないことを示すことになっています。

これで**リスト1**のしていることが，おわかりいただけたでしょうか。他にも多くのサービスリクエストがありますので，マニュアルをご覧ください。

サブルーチンとしての使い方

リスト1はshellから呼び出して使うプログラムでしたが，次に，BASICから呼

リスト4

```
Microware OS-9 Assembler 2.1    00/00/00 00:06:38                 Page 001
isort - OS-9 System Symbol Definitions

00001                                   nam    isort
00002                                   ifp1
00004                                   endc
00005   0000 87CD004C                   mod    msize,name,type,revs,start,size
00006   0021               type         set    sbrtn+objct
00007   0081               revs         set    reent+1
00008   000D 69736F72      name         fcs    /isort/
00009 D 0000               size         equ    .
00010   0012               start        equ    *
00011   0012 EC62                       ldd    2,s
00012   0014 10830001                   cmpd   #1
00013   0018 262A                       bne    error
00014   001A 10AE66                     ldy    6,s
00015   001D               loop0
00016   001D 313E                       leay   -2,y
00017   001F 2721                       beq    exit
00018   0021 AE64                       ldx    4,s
00019   0023               loop1
00020   0023 EC81                       ldd    ,x++
00021   0025 10A384                     cmpd   ,x
00022   0028 2F0A                       ble    skip
00023   002A 3406                       pshs   a,b
00024   002C EC84                       ldd    ,x
00025   002E ED1E                       std    -2,x
00026   0030 3506                       puls   a,b
00027   0032 ED84                       std    ,x
00028   0034               skip
00029   0034 313E                       leay   -2,y
00030   0036 26EB                       bne    loop1
00031   0038 10AE66                     ldy    6,s
00032   003B 313E                       leay   -2,y
00033   003D 10AF66                     sty    6,s
00034   0040 20DB                       bra    loop0
00035   0042               exit
00036   0042 5F                         clrb
00037   0043 39                         rts
00038   0044               error
00039   0044 86FF                       lda    #$ff
00040   0046 1F8A                       tfr    a,cc
00041   0048 39                         rts
00042   0049 90270F                     emod
00043   004C               msize        equ    *
00044                                   end

00000 error(s)
00000 warning(s)
$004C 00076 program bytes generated
$0000 00000 data bytes allocated
$23C5 09157 bytes used for symbols
```

び出すマシン語サブルーチンを紹介しましょう。

リスト４は整数型配列のデータを昇順にソートするプログラムです。

マシン語によってプログラムを書くことで，処理速度が上がるだけでなく，配列の大きさや，引数の数によらないプロシージャを書くことができます。BASIC09 の配列の大きさは可変にできないので，汎用のソーティングプロシージャを BASIC09 で書くことはできません。マシン語では引数の大きさが利用できるので，配列の大きさによらないプロシージャが書けます。引数がシステムスタック上でどのように渡されるかを示したのが**図２**です。

リスト４はバブルソートです。**リスト１**との違いは type が sbrtnになっていることが，まずあげられます。このためデータ領域を rmbで確保しようとしても実行時には確保されません。このプロシジャの作業領域はスタック上にとります。**リスト４**では特に使っていませんが，数バイト以上の領域を必要としたら，スタックを下げて作業領域を作ります。

マシン語をデバッグするためには，普通はデバッガを使います。Ｂレジスタに値を入れて戻るとエラーが生じたものとみなされて，エラー番号を表示して，つまり，Ｂレジスタの値を表示してストップすることを利用してもデバッグができます。

リスト５は，**リスト４**を使ったソートテストです。BASIC09 のプロシジャで行った場合（**リスト６**）と比較してみてください。なお，ISORTB(BASIC09)とISORT(マシン語)の実行時間の比較を**図３**に載せておきます。

OS-９でのマシン語プログラムの書き方は，以上で多少ともおわかりいただけたと思います。例としてあげたプログラムは簡単なものですが，実用性はありますし，改良もしやすいはずです。これを参考にして，多くの方がマシン語を使われることを期待します。

図２　引数の渡され方

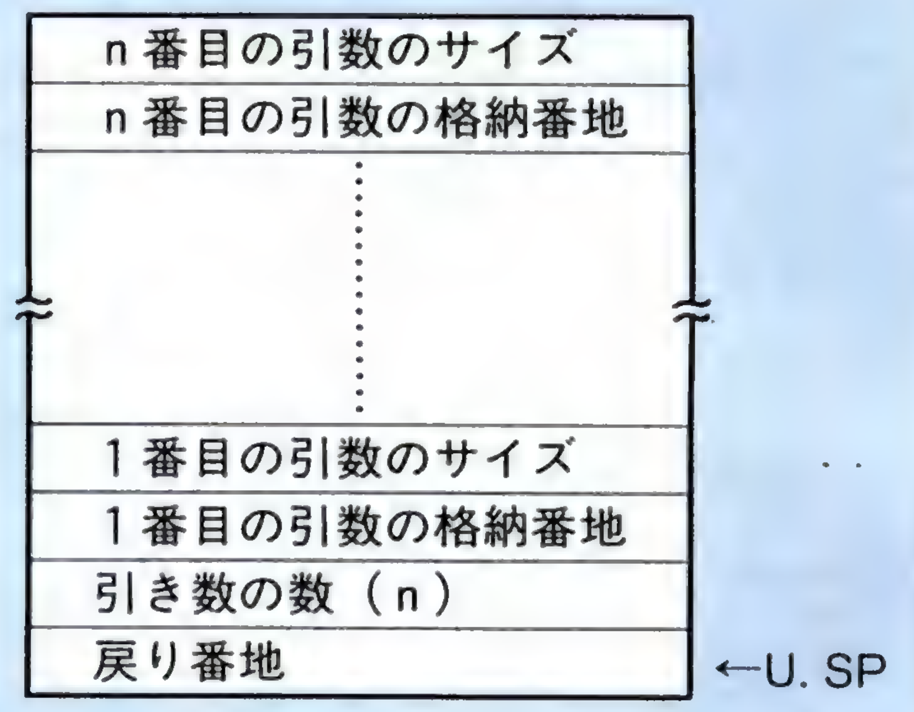


図３　ISORTBとISORTの実行時間の比較

ベンチマーク

プログラム	時間(秒)
ISORTB	1293
ISORT	14.8

$\frac{\text{ISORTB}}{\text{ISORT}}=87.4$

リスト５

```
PROCEDURE stest
 0000     DIM A(1000):INTEGER
 000C     DIM I,J:INTEGER
 0017     PRINT DATE$
 001A     FOR J=1 TO 10
 002A       FOR I=1 TO 1000
 003B         A(I)=1000-I
 004B       NEXT I
 0056     NEXT J
 0061     PRINT DATE$
 0064     FOR J=1 TO 10
 0074       FOR I=1 TO 1000
 0085         A(I)=1000-I
 0095       NEXT I
 00A0       RUN ISORT(A)
 00AA     NEXT J
 00B5     PRINT DATE$
 00B8     END
```

リスト６

```
PROCEDURE stestb
 0000     DIM A(1000):INTEGER
 000C     DIM I:INTEGER
 0013     FOR I=1 TO 1000
 0024       A(I)=1000-I
 0034     NEXT I
 003F     PRINT DATE$
 0042     RUN ISORTB(A)
 004C     PRINT DATE$
 004F     END
PROCEDURE isortb
 0000     PARAM a(1000):INTEGER
 000C     DIM i:INTEGER
 0013     DIM j:INTEGER
 001A     FOR i=999 TO 1 STEP -1
 0031       FOR j=1 TO i
 0042         IF a(j)>a(j+1) THEN
 0058           t=a(j)
 0064           a(j)=a(j+1)
 0076           a(j+1)=t
 0086         ENDIF
 0088       NEXT j
 0093     NEXT i
 009E     END
```


バックナンバー情報(Vol. 1 〜'84 ８月号まで)

Vol.1〜6は品切れ。Vol.1〜3は別冊「FILES Oh !FM」として合本。
Vol. 7　特集/ディスク活用術，オールRAM活用法，碁子の住所録，他
４月号　特集/虫取り専科，使い勝手のよいマシン語モニタ，FM-11 BS-AD2，アドベンチャゲーム開発ツール，他
５月号　特集/日本語ワープロ徹底研究，FM-7を解析する，レイトレーシングを用いた３次元グラフィックス，他
６月号　特集/エンジョイ!!FM音楽ゴコロ，FM-77とFM-NEW7，FM-11サブシステムの研究，6809機械語マスタ'84，他
７月号　特集/おいしいプリンタ大研究，BDOSコール入門，アニメキャラクタプログラム，FM-7を解析する，他
８月号　特集/この夏，ゲームで熱く!，FM-77試用レポート，BASICプログラムコンプレッサ，オンメモリBASICチェックサム，他

●バックナンバーご希望の方は，最寄りの書店に申し込むか，常備店(８月号P. 148参照)をご利用ください。

OS-9 逆アセンブラ V1.0

for OS-9

Simple Logic

OS-9の6809機械語モジュールを逆アセンブルするプログラムを，BASIC 09を使って作りましたので紹介します。今回は，バージョン1.0として発表します。最終的には，OS-9のアセンブラ(Asm)にかかるようなアセンブラソースプログラムを出力できるようにしたいと思っています。いろいろ付加したい機能がありますが，将来，それらを拡張することを前提として作りました。とりあえず今回のバージョンでも，システムプログラムの解析には使えると思います。

概要

この逆アセンブラは，オブジェクトをディスクファイルから入力するようにしてあります。これは，逆アセンブラ本体がかなり大きいため，メモリ上のモジュールを逆アセンブルするように作ると，逆アセンブラ本体と同時にメモリ上に置けないような大きなプログラムが逆アセンブルできないからです。したがって，メモリ上のOS-9のシステムモジュール(Os9p1，Os9p2，SCFMan，RBFManなど)を逆アセンブルしたい場合は，一度，SAVEコマンドを使って，ディスク上のファイルにしておく必要があります。

出力されるソースプログラムにはラベルを付けられますので，作業領域としてラベルテーブルが必要です。この逆アセンブラでは，これもディスク上のファイルを使っています。したがって，実行中に頻繁にディスクをアクセスするので，逆アセンブルの速さは若干遅くなっています。また，このラベルテーブルは，プログラムを簡単にするため，ラベルの名前を入れるのではなく，64KBのファイルをメモリ空間に対応させて，各バイトのデータ内容で，そのアドレスに対応するラベルを示す構造にしています（「プログラムの説明」参照）。したがって，ラベルファイルでディスクを64KB使うことになります。出力形式は，アセンブラのソースプログラム形式と，アドレス，機械語の付いたアセンブルリスト形式を選ぶことができます。当然パスリストで出力先を指定するので，ファイルや，プリンタへも出力できます。ただし，ソース形式で出力しても，今回のバージョンではそのままでは，アセンブラにかけることはできません。それは，プログラムの範囲外のアドレスを持つラベルが定義されないためです。

また，エクステンドアドレッシングや，ダイレクトアドレッシング(JMP，JSRを除く）は，オペランドには表れますが，ラベルフィールドには出ないようにしてあります。これは，OS-9のプログラムは，すべてリロケータブルであるはずだからです。将来は，これらのラベルは，出力プログラムの先頭でEQU定義されるようにしたいと思います。

そのほか，未定義命令はFCBで表され，モジュール定義のためのMOD命令，EMOD命令も出されます。OS-9のシステムコール(SWI 2)はos9疑似命令で出力され，そのときOS-9のシステムコールのラベルがつくようにしてあります。

プログラムの入力

このプログラムは，テキストの状態で約30KB，パックすると約半分になります。OS-9レベル1を使っている人は，いっぺんに入力することは不可能なので，各プロシジャごとに打ち込んで，セーブしてからパックしていきます。そして全部打ち込んだら，MERGEコマンド(これはBASIC09の命令ではありません)でパックされた各モジュールを1つにまとめ，disasという名のファイルにしておきます。これは，「複数のモジュールがまとまったモジュールは，その中で一番はじめに実行されるモジュール名をファイル名としておかないといけない」からです。もしdisasというファイル名では都合が悪ければ，disasといういちばんはじめのプロシジャ名を，都合のよいファイル名と同じ名前に変える必要があります(BASIC09のRENAMEコマンドを使う)。

FM-11でOS-9レベル2を使っている方は，いっぺんに入力できます。

使用方法

図1に実行例を示します。

disas⏎として逆アセンブラを起動するとメニュー形式で聞いてきます。

①はじめにグリーティングメッセージが表示されたあと，

target file name：

と聞いてくるので，逆アセンブルしたいモジュールの入っているファイルのパスリストを入力します。

②次に，

select label_initializing mode

1…create new file

2…update existing file

3…re－use existing file

：

と聞いてきます。これは，ラベルファイルをどう初期化するかを指定します。

1は，新しいファイルをつくって，初期化します。

2は，すでに存在するファイルを初期化せずに使います。

3は，すでに存在しているファイルを，初期化して使います。

③すると次に，

label file name：

と聞いてくるので，ラベルテーブルとして使うファイルのパスリストを入力します。

初期化の指定があれば（②で1か3を指定したとき）

Initializing label file, wait a moment!

と表示して，ラベルファイルの初期化を行います。

④次に，

select pass＿type

－1…exit to OS-9

0…simple disassembler（no label）

1…label－registration

2…label－registration＆reference

3…label＿reference

：

と聞いてきます。

－1は逆アセンブラを終了します。

0はラベル変換を行わない逆アセンブラです。

1は，ラベルの登録だけを行うパスで，ラベルフィールドにはラベルが出ませんが，オペランドフィールドにはラベルが出ます。

2はラベル登録をしながら参照も行い，ラベルフィールドにもラベルを出します。ただしラベルが前方参照になっているものは，もちろん表示されません。

3は，すでに1か2でラベル登録をした後で最終的にリストを出力するときに使うとよいでしょう。登録はせず参照だけですが，ラベルフィールド，オペランドフィールドともにラベルが出力されます。

なお，ラベルの形式については「プログラムの説明」で説明します。

⑤次に，

select output－path(path＿name,null‥console)：

と表示されます。これは，逆アセンブルリストの出力パスの指定です。リターンのみを入力すると標準出力パスが指定されます（通常はCRT）。なお，今まで示した表示や今後述べる逆アセンブルリスト以外の出力もすべて標準出力パスに出力されています。

⑥次に

select output－mode

0…only source code

1…with address ＆ machine code

：

と表示されます。

0を指定すると，アセンブラのソースプログラム形式の出力となります。

1を指定すると，アドレス，機械語の付いた，アセンブルリスト形式で出力されるようになります。

⑦最後に逆アセンブルの範囲を聞いてきます。

start－address（hex）：

スタートアドレスは，16進（「$」は不要）で入力します。リターンのみを打つとスタートアドレスは$0000となります。

end－address（hex）：

エンドアドレスも16進で入力します。リターンのみで$FFFFとなります。つまりモ

図1　使用例（色文字を入力）

```
OS9:disas↵

♣♣♣♣♣♣♣♣♣♣ OS-9 module-disassembler v.1.0 ♣♣♣♣♣♣♣♣♣♣

target file name: /d0/cmds/echo↵
select label_initializing mode
        1...create new file
        2...update existing file
        3...re-use existing file
:1↵
label file name : label↵
Initializing label file ,wait a moment!
select pass-type
          -1...exit to OS-9
           0...simple disassembler(no label)
           1...label-registration
           2...label-registration & reference
           3...label_reference
:0↵
select output-path(path_name,null..console):↵
select output-mode
           0...only source code
           1...with address & machine code
:1↵
start-address(hex):↵
end-address(hex)  :↵
***************************
    module type= prog
    language    = 6809
    attribute   = re-entrant
    revision    = $1
***************************
0000 87 cd 00 22 00 0d 11 81 0a 00 12 01 c2   mod $0022,$000d,$11,$81,$0012,$01c
2  .M.".......B
000d 45                             FCB   $45           'E' E
000e 63 68                          com   8,s               ch
0010 ef 05                          stu   5,x               o.
0012 1f 02                          tfr   d,y               ..
0014 86 01                          lda   #$01              ..
```

```
0016 10 3f 8c                  os9    I$WritLn         .?.
0019 25 01                     bcs    $001c            %.
001b 5f                        clrb                    _
001c 10 3f 06                  os9    F$Exit           .?.
*
001f f5 ff 9a                  emod                    u.
*
select pass-type
         -1...exit to OS-9
          0...simple disassembler(no label)
          1...label-registration
          2...label-registration & reference
          3...label_reference
:1
select output-path(path_name,null..console):
select putput-mode
          0...only source code
          1...with address & machine code
:1
start-address(hex):
end-address(hex)  :
***************************
    module type= prog
    language   = 6809
    attribute  = re-entrant
    revision   = $1
***************************
0000 87 cd 00 22 00 0d 11 81 0a 00 12 01 c2   mod PP_001f+3,PD_000d,$11,$81,PP_0
012,$01c2  .M.'.......B
000d 45                        FCB    $45          'E' E
000e 63 68                     com    8,s              ch
0010 ef 05                     stu    5,x              o.
0012 1f 02                     tfr    d,y              ..
0014 86 01                     lda    #$01             ..
0016 10 3f 8c                  os9    I$WritLn         .?.
0019 25 01                     bcs    PP_001c          %.
001b 5f                        clrb                    _
001c 10 3f 06                  os9    F$Exit           .?.
*
001f f5 ff 9a                  emod                    u.
*
select pass-type
         -1...exit to OS-9
          0...simple disassembler(no label)
          1...label-registration
          2...label-registration & reference
          3...label_reference
:3
select output-path(path_name,null..console):
select output-mode

          0...only source code
          1...with address & machine code
:1
start-address(hex):
end-address(hex)  :
***************************
    module type= prog
    language   = 6809
    attribute  = re-entrant
    revision   = $1
***************************
0000 87 cd 00 22 00 0d 11 81 0a 00 12 01 c2   mod PP_001f+3,PD_000d,$11,$81,PP_0
012,$01c2  .M.'.......B
000d 45                PD_000d FCB    $45          'E' E
000e 63 68                     com    8,s              ch
0010 ef 05                     stu    5,x              o.
0012 1f 02             PP_0012 tfr    d,y              ..
0014 86 01                     lda    #$01             ..
0016 10 3f 8c                  os9    I$WritLn         .?.
0019 25 01                     bcs    PP_001c          %.
001b 5f                        clrb                    _
001c 10 3f 06          PP_001c os9    F$Exit           .?.
*
001f f5 ff 9a          PP_001f emod                    u.
*
select pass-type
         -1...exit to OS-9
          0...simple disassembler(no label)
          1...label-registration
          2...label-registration & reference
          3...label_reference
:-1
OS9:
```

ジュールの全範囲を逆アセンブルしたければ続けて２回リターンを打てばよいのです。

⑧すると，CRT にモジュールの属性が表示され(標準エラー出力パス)，その後，⑦で指定したスタートアドレスから逆アセンブルリストまたはソースプログラムが，指定した情報をもとに⑤で指定したパスに出力されます。出力パスに CRT を指定した場合，ターミナルモードがポーズ指定になっていると，25行表示するたびに出力が停止します。これがいやなときは，あらかじめ shell の入力待ちで tmode－pause を実行しておいてください。逆アセンブルは，①で指定したオブジェクトモジュールファイルの EOF (end of file) を検出するか，⑦で指定したエンドアドレスを超えるまで続けられます。

逆アセンブルが終了すると，④のパスタイプの指定に戻ります。

プログラムの説明

▶ラベルの出力形式

ラベルは次のような形式です。

○○_××……………………(1)

○○_△△△△……………(2)

はじめの２文字はラベルの属性を表します。ラベルの属性は，図２のように全部で６種に分類されています。××および△△△△はラベルのアドレス (16進) です。ダイレクトモードは(1)のように２桁，その他は４桁です。

▶ラベルの選択出力

オペランドフィールドにはすべての種類のラベルが出力されますが，ラベルフィールドには，DP，PD，PP の属性を持つラベ

ルのみが出力されます。これはOS-9のモジュールはすべてリロケータブルであり，モジュール内にワークエリアが存在しないからです。DPの属性を持つラベルを出力するのは，OS-9では，ダイレクトページレジスタにプログラム先頭のページを入れておいてダイレクトモードのjmp, jsr命令を使うことがよくあるからです。

▶ラベルファイル

ラベルファイルは64KBの大きさで，逆アセンブル対象のモジュールからみたアドレス空間に対応するラベルファイル内のデータ位置（バイト単位）に，属性を表す情報が書き込まれるようになっています。図3に各属性のビット対応を示します。

▶命令の途中にラベルがある場合

命令の先頭アドレスにラベルのアドレスが対応しているのが普通ですが，データエリアが逆アセンブルされている場合や，プログラムトリックのある場合で命令の途中にラベルが定義されてしまっている場合は，その命令の直後にEQU疑似命令で図4のように出力されます。

▶プログラムの構造

図2　ラベルの属性表示

表示	アドレッシングモード	種　別	例
DD	ダイレクトアドレッシング	データ	sta ＜DD_00
DP		プログラム	jsr ＜DP_10
ED	イミディエイトまたはエクステンデッドアドレッシング	データ	ldd #ED_FFFF lda ＜EP_0100
EP		プログラム	jmp ＜EP_2000
PD	プログラムカウンタリラティブアドレッシング	データ	ldb PD_0200,pcr
PP		プログラム	lbsr PP_1100

図3　ラベルファイル内の属性データのビット対応

MSB 7	6	5	4	3	2	1	0 LSB
DD	DP	ED	EP	PD	PP	╳	╳

図4　命令の途中にラベルがある場合の出力例

```
0064 10 83 00 01        PP_0064 cmpd   #ED_0001          ....
0068 22 41                      bhi    PP_00ab           "A
006a 33 22                      leau   2,y               3"
           :
00a8 c6 2e              PP_00a8 ldb    #$2e         '.'  F.
00aa 8e c6 38                   ldx    #ED_c638          .F8
00ad                    PP_00ab EQU    *-2        ***
00ad 43                         coma                     C
```

図5に各プロシジャの関係を示します。

各プロシジャの働きは，次に述べるようになっています。

●**disas**

この逆アセンブラのメインルーチンで，メニュー形式のコマンド入力と，逆アセンブルのメインループから成っています。

●**disasm_1_line**

1命令だけの逆アセンブルを行うプロシジャで，ニモニックとオペランド・その他を返します。

● **word_immediate, byte_immediate, direct_page indexed, extended, short_branch, long_branch**

それぞれ，各アドレッシングモードのオペランドを作成するルーチンで，必要に応じてlabel_managerを呼んでラベルの登録を行います。

● **os9**

OS-9のシステムコール命令(SWI2)のオペランドを作成します。これは，OS9Defsに定義されたものと同一です。

● **module_header**

モジュールヘッダの内容を解析して，必要なパラメータはラベル登録します。また，そのモジュールの属性を標準エラー出力パスに出力します。

● **emod**

モジュールの終わりを示すemod命令の処理です。CRCが正しいかどうかのチェックは行っていません。

● **opcode_main**

オペコードを解析してニモニックとアドレッシングモードを返します。実際にはプリバイトの有無を調べopcode_normal, opcode_pre_10, opcode_pre_11に処理を振り分けるだけです。

● **opcode_pre_10, opcode_pre_11**

それぞれプリバイトが$10, $11の命令のオペコードを解析してニモニックと，アドレッシングモードを返します。

● **label_initialize**

ラベルファイルのオープンおよびイニシャライズを行います。イニシャライズは，すべてのデータバイトをすべて$00に書き換えるだけです。

● **label_reference**

与えられたアドレスからラベルファイルをサーチして次のラベルを返します。

● **label_manager**

ラベルの登録を行います。登録は属性バイトのORをとっていくので，同一アドレスに異なるタイプのラベルが対応していても独立に登録されます。

● **find_attr**

属性バイトの「1」のビットをサーチするプログラムで，label_reference から呼ばれます。

● **print_source**

逆アセンブル結果を，指定されたフォーマットで指定されたパスに出力します。

● **make_mcasc**

マシンコードとその ASCII コードを文字列にします。

● **make_2hex, make_4hex**

BASIC09 には HEX$関数がないので，これらのプロシジャを使っています。

● **get_next_byte, get_next_word**

オブジェクトモジュールファイルから，それぞれ1バイト，2バイトを取り出すプロシジャです。

● **byte_to_int**

1バイトを2の補数表示とみたときの値を計算します。

● **int_to_real**

この逆アセンブラプログラム全体を通じて，アドレスは整数型の変数に入れていますが，整数型は2の補数表示なので，アドレスの比較の際に困ります。そこでこのプロシジャは，整数型の値を，0～65535の正の数と見た場合の値に変換します。

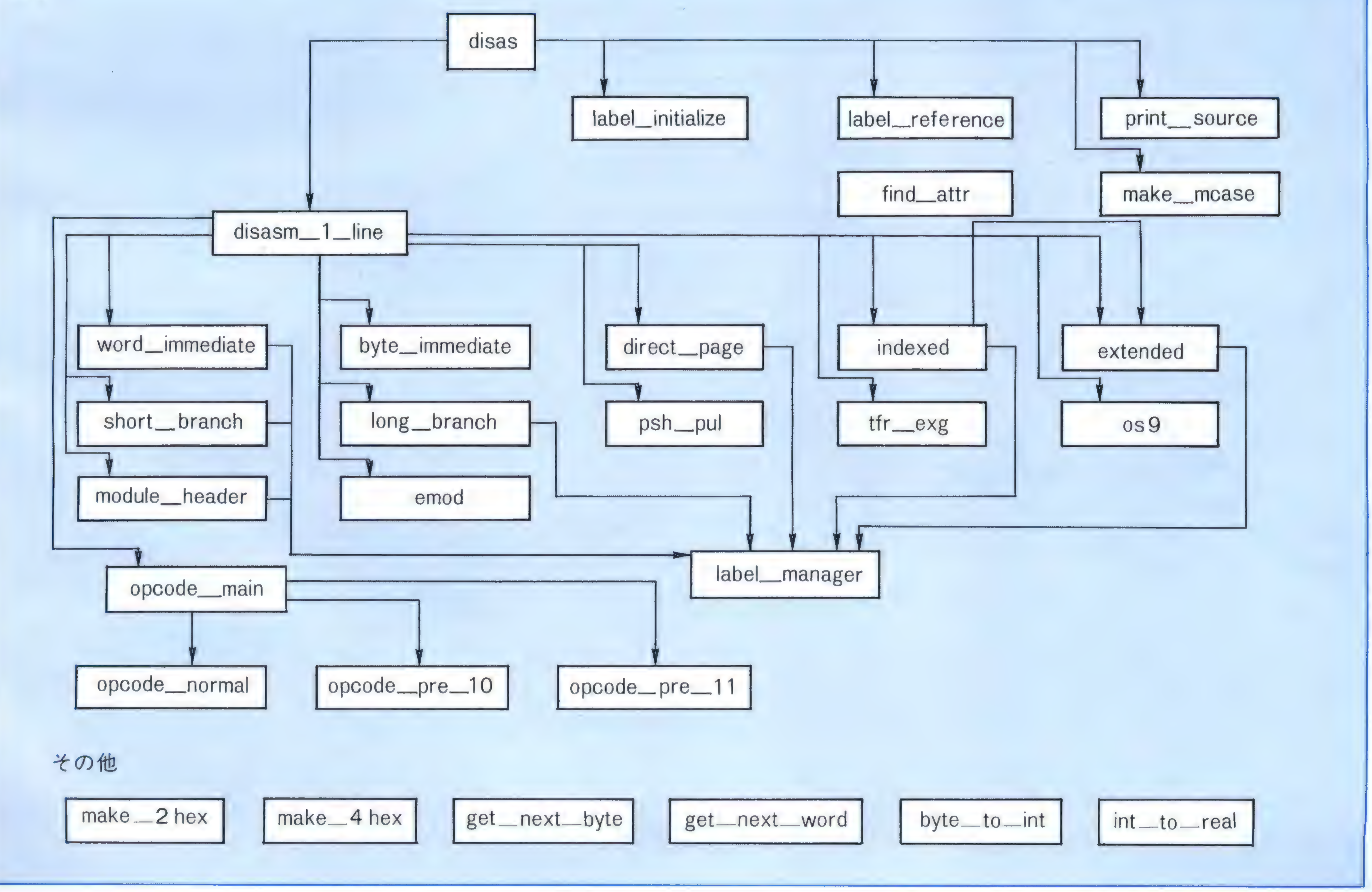
さいごに

最後に OS-9と BASIC09 について，苦言をいくつか述べておきます。

私は，この逆アセンブラの作成にあたって FM-11の OS-9レベル2 (V1.1 L2.1) を使ったのですが，これには若干のバグがあるようです。どうもプログラムのセーブ中に Control-C を打つと，ディスクとディレクトリが破壊されることがあるようです。私は，このプログラム作成中に2回被害に遭いました。1回目は，一晩かけてディレクトリを復元するハメになりました（おかげで OS-9のディスクのフォーマットがよくわかりました)。2回目は，この原稿を書いているときで，原稿提出が1日遅れてしまいました。使っている人は注意したほうがいいですよ。V1.1 L2.2 ではこのバグはとれているようです。

今回のプログラムは，BASIC09 を使ってはじめて書いたまともなプログラムなので，いくつか，BASIC09 について気付いた点を述べておきます。

図5



まず，BASIC09のエディタはちょっと貧弱すぎるように思います。特に行のコピー，ムーブがないのはつらい。また，行内の編集機能としてCコマンドしかないのも苦しい。プログラムの変更はOS-9のエディタ（EDIT）を使ったほうがよいようです。

変数がすべてローカル変数だということも，プログラムのモジュール化のためにはよいのですが，大きなプログラムだと引数がどうしても多くなってしまいます。構造体を使ってグローバル変数を1つにまとめたほうがよかったかもしれません。どうせインタプリタなのですから，たとえば，同一ファイル内のプロシジャにはグローバル変数が使えるようにしておいてほしかったと思います。

CASE文（あるいはSWITCH文）がないのにも困りました。行番号嫌悪症の私は，ON～GOSUB文は使いたくなかったのでIF文を使ったのですが，そうするとネスティングがどんどん深くなり，リストが見にくくなってしまいました。

また，Shellから引き数が渡せるようにしてほしかったと思います。

以上，ロクなプログラムも書けないくせに文句ばかり書きましたが，もし，バージョンアップされるのでしたら，ぜひ改良してほしいと思っています。

さいごのさいごに

本プログラムは，まだ改良したい部分がたくさんあります。不充分な点はどうかご容赦ねがいます。今後，より完全な逆アセンブラをめざしてバージョンアップしていきたいと思います。

バグはほぼ退治したと思いますが，もしバグを発見された方は，編集部まで郵便などでご通報ください。

逆アセンブラリスト

© 1984 Simple Logic

```
PROCEDURE disas
 0000      DIM obj_path,adr:INTEGER; w_opc:BOOLEAN; c_cnt:INTEGER; m_code
             (20):BYTE
 0022      DIM opc_typ:INTEGER; end_flag:BOOLEAN; l$,ll$:STRING[8]; m$
             :STRING[5]; op$,com$:STRING[50]
 0058      DIM mod_siz,mod_nam,exec_ofs,st_siz:INTEGER
 006B      DIM typ_lng,atr_rev,head_par:BYTE
 007A      DIM not_6809:BOOLEAN
 0081      DIM loc$:STRING[4]
 008D      DIM file_name$:STRING[50]
 0099      DIM address,label_address:INTEGER
 00A4      DIM n:INTEGER; hex$:STRING[50]
 00B6      DIM term:BOOLEAN
 00BD      DIM label_file$:STRING[32]; label_fp:INTEGER
 00CF      DIM ascii$:STRING[20]
 00DB      DIM pass_type,label_init_mode:INTEGER
 00E6      DIM find,command_exit:BOOLEAN
 00F1      DIM out_path$:STRING[50]
 00FD      DIM out_mode:INTEGER
 0104      DIM end_adr:INTEGER
 010B      DIM real_adr,real_end_adr,real_address,real_label_address:REAL
 011E      DIM out_fp:INTEGER
 0125      DIM l_attr,e_attr:BYTE
 0130      DIM emod_adr:INTEGER
 0137      PRINT '********** OS-9 module-disassembler v.1.0 **********'
 016F      PRINT
 0171      INPUT 'target file name: ',file_name$
 018B      OPEN #obj_path,file_name$:READ
 0197      (* label file initializing mode *)
 01B9      label_init_mode=-1
 01C1      WHILE label_init_mode<1 OR label_init_mode>3 DO
 01D4        PRINT 'select label_initializing mode'
 01F6        PRINT '        1...create new file'
 0214        PRINT '        2...update existing file'
 0237        PRINT '        3...re-use existing file'
 025A        INPUT ':',label_init_mode
 0263      ENDWHILE
 0267      INPUT 'label file name : ',label_file$
 0281      RUN label_initialize(label_file$,label_fp,label_init_mode)
 0295      mod_siz=$FFFF
 029D      emod_adr=$FFFF
 02A5      command_exit=FALSE
 02AB      WHILE command_exit=FALSE DO
 02B6        (* pass type selection *)
 02CF        pass_type=-2
 02D7        WHILE pass_type<-1 OR pass_type>3 DO
 02EB          PRINT 'select pass-type'
 02FF          PRINT '          -1...exit to OS-9'
 031D          PRINT '           0...simple disassembler(no label)'
 034C          PRINT '           1...label-registration'
 0370          PRINT '           2...label-registration & reference'
 03A0          PRINT '           3...label_reference'
 03C1          INPUT ':',pass_type
 03CA        ENDWHILE
 03CE        IF pass_type=-1 THEN command_exit=TRUE
 03E0        ELSE (* not command exit *)
 03F9          INPUT 'select output-path(path_name,null..console):',out_path$
 042D          IF out_path$='' THEN (* standard output path *)
 0453            out_fp=1
 045A          ELSE (* user-defined path *)
 0474            CREATE #out_fp,out_path$:WRITE
 0480          ENDIF
 0482          (*.source output path *)
 049A          out_mode=-1
 04A2          WHILE out_mode<0 OR out_mode>1 DO
 04B5            PRINT 'select output-mode'
 04CB            PRINT '           0...only source code'
 04ED            PRINT '           1...with address & machine code'
 051A            INPUT ':',out_mode
 0523          ENDWHILE
 0527          (* address range *)
 053A          INPUT 'start-address(hex):',loc$
 0555          IF loc$='' THEN
 0561            adr=0
 0568          ELSE
 056C            adr=VAL('$'+loc$)
 057A          ENDIF
 057C          INPUT 'end-address(hex)   :',loc$
 0597          IF loc$='' THEN
 05A3            end_adr=$FFFF
 05AB          ELSE
 05AF            end_adr=VAL('$'+loc$)
 05BD          ENDIF
 05BF          RUN int_to_real(end_adr,real_end_adr)
 05CE          end_flag=FALSE
 05D4          label_address=adr-1
 05DF          find=FALSE
 05E5          l$=''
 05EC          l_attr=$00
 05F4          e_attr=$4C ¥(* ip,pd,pp *)
 060A          (* **** main loop ***** )
 0622          WHILE end_flag=FALSE DO
 062D            ll$=''
 0634            address=adr
 063C            RUN int_to_real(address,real_address)
 064B            IF (pass_type=2 OR pass_type=3) AND find=FALSE THEN (*
                   search next label *)
 067A              RUN label_reference(label_fp,label_address,l$,find,
                     l_attr,e_attr,emod_adr)
 06A2            ENDIF
 06A4            IF find THEN (* more labels *)
 06BD              RUN int_to_real(address,real_address)
 06CC              RUN int_to_real(label_address,real_label_address)
 06DB              WHILE real_address>real_label_address AND find DO (*
                     mysterious label *)
 0701                RUN make_4hex(address,loc$)
 0710                RUN make_mcasc(m_code,c_cnt,hex$,ascii$)
 0729                find=FALSE
```

```
072F            ll$=''
0736            hex$=''
073D            m$='EQU'
0747            op$='*-'+TRIM$(STR$(address-label_address))
075A            com$='***'
0764            ascii$=''
076B            RUN print_source(loc$,hex$,l$,m$,op$,com$,ascii$,
                 pass_type,out_fp,out_mode,term)
07A7            RUN label_reference(label_fp,label_address,l$,find
                 ,l_attr,e_attr,emod_adr)
07CF            RUN int_to_real(label_address,real_label_address)

07DE          ENDWHILE
07E2          IF address=label_address THEN (* normal label process *)
0808            ll$=l$
0810            find=FALSE
0816          ENDIF
0818        ELSE (* no label *)
0829          ll$=''
0830        ENDIF
0832        m$=''
0839        op$=''
0840        com$=''
0847        IF pass_type=0 THEN (* no label...label_fp=0 *)
086D          RUN disasm_1_line(obj_path,0,adr,term,w_opc,c_cnt,m_code
               ,opc_typ,end_flag,m$,op$,com$,mod_siz,mod_nam
               ,exec_ofs,st_siz,typ_lng,atr_rev,head_par,not_6809
               ,emod_adr)
08D9        ELSE (* label processing *)
08F2          IF pass_type=2 OR pass_type=1 THEN (* resist *)
0910            RUN disasm_1_line(obj_path,label_fp,adr,term,w_opc
                 ,c_cnt,m_code,opc_typ,end_flag,m$,op$,com$,
                 mod_siz,mod_nam,exec_ofs,st_siz,typ_lng,atr_rev
                 ,head_par,not_6809,emod_adr)
097E          ELSE (* pass_type=3 *)
0992            RUN disasm_1_line(obj_path,1,adr,term,w_opc,c_cnt
                 ,m_code,opc_typ,end_flag,m$,op$,com$,mod_siz
                 ,mod_nam,exec_ofs,st_siz,typ_lng,atr_rev,head_par
                 ,not_6809,emod_adr)
09FE          ENDIF
0A00        ENDIF
0A02        IF end_flag=FALSE THEN
0A0D          RUN make_4hex(address,loc$)
0A1C          RUN make_mcasc(m_code,c_cnt,hex$,ascii$)
0A35          RUN print_source(loc$,hex$,ll$,m$,op$,com$,ascii$,pass_type
               ,out_fp,out_mode,term)
0A71        ENDIF
0A73        IF real_address>real_end_adr THEN (* address over *)
0A91          end_flag=TRUE
0A97        ENDIF
0A99      ENDWHILE
0A9D      IF out_fp>1 THEN
0AA9        CLOSE #out_fp
0AAF      ENDIF
0AB1    ENDIF
0AB3  ENDWHILE
0AB7  CLOSE #obj_path
0ABD  CLOSE #label_fp
0AC3  END
PROCEDURE opcode_main
0000    (* ♥♥♥♥ opcode dispatch ♥♥♥♥
001C    (* 1 byte instruction ... opcode_normal
0043    (* $10 prebyte inst.  ... opcode_pre_10
0069    (* $11 prebyte inst.  ... opcode_pre_11
008F    PARAM object_path_no,address:INTEGER; term:BOOLEAN; opcode_type
         :INTEGER; end_flag:BOOLEAN; mnemonic$:STRING[5]; word_opcode
         :BOOLEAN; machine_code(20):BYTE
00C8    PARAM emod_adr:INTEGER
00CF    DIM opcode,opcode2:BYTE
00DA    IF emod_adr=address THEN (* end of module *)
00F9      opcode_type=-2
0101      mnemonic$='emod'
010C      RUN get_next_byte(object_path_no,address,opcode,end_flag)

0125      IF end_flag THEN  ¥ ENDIF
0130      RUN get_next_byte(object_path_no,address,opcode2,end_flag
           )
0149      IF end_flag THEN  ¥ ENDIF
0154      machine_code(1)=opcode
015F      machine_code(2)=opcode2
016A      word_opcode=TRUE
0170    ELSE
0174      RUN get_next_byte(object_path_no,address,opcode,end_flag)

018D      IF end_flag THEN END  ¥ ENDIF
0199      machine_code(1)=opcode
01A4      IF opcode=$10 OR opcode=$11 OR opcode=$87 THEN
01C1        RUN get_next_byte(object_path_no,address,opcode2,end_flag
             )
01DA        IF end_flag THEN END  ¥ ENDIF
01E6        machine_code(2)=opcode2
01F1        word_opcode=TRUE
01F7        IF opcode=$10 THEN
0204          RUN opcode_pre_10(opcode2,opcode_type,mnemonic$)
0218        ELSE (* opcode=$11 *)
022C          IF opcode=$11 THEN
0239            RUN opcode_pre_11(opcode2,opcode_type,mnemonic$)
024D          ELSE
0251            IF opcode2=$CD THEN
025E              opcode_type=-1 ¥(* sync byte *)
0275              mnemonic$='mod'
027F            ELSE
0283              opcode_type=0
028A            ENDIF
028C          ENDIF
028E        ENDIF
0290      ELSE
0294        word_opcode=FALSE
029A        RUN opcode_normal(term,opcode,opcode_type,mnemonic$)
02B3      ENDIF
02B5    ENDIF
02B7    END
PROCEDURE opcode_normal
0000    PARAM term:BOOLEAN
0007    PARAM opcode:BYTE; opcode_type:INTEGER; mnemonic$:STRING[5]

001F    DIM high,low:BYTE
002A    high=opcode/$10
0036    low=opcode-high*$10
0046    IF high>=$08 THEN
0053      opcode_type=MOD(high-8,4)+2
0064      IF low<=$0B THEN
0071        IF low=3 THEN (* addd,subd *)
008B          mnemonic$=MID$('subdaddd',4*((high-8)/4)+1,4)
00AC          IF opcode_type=2 THEN
00B8            opcode_type=1
00BF          ENDIF
00C1        ELSE (* sub,cmp,sbc,and,bit,ld,st,eor,adc,or,add + a,b *)
00F8          mnemonic$=TRIM$(MID$('subcmpsbc***andbitld st eoradcor add'
               ,3*low+1,3))
0130          mnemonic$=mnemonic$+MID$('ab',(high-8)/4+1,1)
014C        ENDIF
014E      ELSE (* cmpx,ldd,bsr,jsr,std,ldx,ldu,stx,stu *)
017B        IF opcode=$8D THEN (* bsr *)
0190          mnemonic$='bsr'
019A          opcode_type=6 ¥(* relative(short) *)
01B6        ELSE
01BA          mnemonic$=TRIM$(MID$('cmpxldd jsr std ldx ldu stx stu '
               ,4*((high-8)/4+(low-$0C)*2)+1,4))
01FF          IF opcode_type=2 THEN
020B            opcode_type=1
0212          ENDIF
0214        ENDIF
0216      ENDIF
0218      IF opcode=$87 OR opcode=$C7 OR opcode=$CD OR opcode=$8F OR
           opcode=$CF THEN
0245        opcode_type=0 ¥(* undefined *)
025B      ENDIF
025D    ELSE (* high<$8 *)
026D      IF high=0 OR high>=4 THEN (* neg,com,lsr,ror,asr,asl/lsl,rol,dec,in
,tst,jmp,clr *)
02B8        IF high=0 THEN opcode_type=3
02CA        ELSE
02CE          IF high=4 OR high=5 THEN opcode_type=10 ¥(* inherent *)
02F5          ELSE (* high=6,7 *)
0306            opcode_type=high-2 ¥(* indirect,extend *)
0326          ENDIF
0328        ENDIF
032A        mnemonic$=MID$('neg******comlsr***rorasrlslroldec***inctstjmpclr'
             ,3*low+1,3)
036D        IF high=4 OR high=5 THEN
0380          mnemonic$=mnemonic$+MID$('ab',high-3,1)
0396        ENDIF
0398        IF low=$0E THEN
03A5          term=TRUE
03AB        ENDIF
03AD      ELSE (* high=1,2,3 *)
03C0        IF high=2 THEN (* branch Bcc *)
03DB          opcode_type=6 ¥(* short branch *)
03F4          mnemonic$='b'+MID$('rarnhilscccsneeqvcvsplmigeltgtle'
               ,low*2+1,2)
042B        ELSE
042F          IF high=1 THEN (* nop,sync,lbra,lbsr,daa,orcc,andcc,sex,exg,tfr
*)
046D            IF low=6 OR low=7 THEN
0480              opcode_type=7 ¥(* long relative *)
```

```
049A              ELSE
049E                IF low=$0A OR low=$0C THEN
04B3                  opcode_type=2 ¥(* byte immediate *)
04CE                ELSE
04D2                  IF low=$0E OR low=$0F THEN
04E7                    opcode_type=9 ¥(* exg,tfr *)
04FB                  ELSE
04FF                    opcode_type=10 ¥(* inherent *)
0514                  ENDIF
0516                ENDIF
0518              ENDIF
051A              mnemonic$=TRIM$(MID$('**********nop  sync **********lbra lbsr
*****daa  orcc *****andccsex  exg  tfr  '
                  ,5*low+1,5))
057E            ELSE (* high=3 *)
058D              IF low<=3 THEN
0599                opcode_type=4 ¥(* indirect *)
05AE              ELSE
05B2                IF low<=7 THEN
05BE                  opcode_type=8 ¥(* psh,pul *)
05D2                ELSE
05D6                  IF low=$0C THEN
05E3                    opcode_type=2 ¥(* byte immediate(cwai) *)
0604                  ELSE
0608                    opcode_type=10 ¥(* inherent *)
061D                  ENDIF
061F                ENDIF
0621              ENDIF
0623              mnemonic$=TRIM$(MID$('leaxleayleasleaupshspulspshupulu****rts
abx rti cwaimul ****swi '
                  ,4*low+1,4))
0677            ENDIF
0679          ENDIF
067B        ENDIF
067D      ENDIF
067F      IF LEFT$(mnemonic$,1)='*' THEN (* undefined *)
069D        opcode_type=0
06A4      ENDIF
06A6      IF opcode=$39 OR opcode=$3B OR opcode=$20 OR opcode=$16 THEN

06CB        term=TRUE
06D1      ENDIF
06D3      END
PROCEDURE opcode_pre_10
0000      PARAM opcode:BYTE; opcode_type:INTEGER; mnemonic$:STRING[5]

0018      DIM high,low:BYTE
0023      high=opcode/$10
002F      low=opcode-high*$10
003F      IF low>0 THEN
004B        low=low-1
0056        IF high<=7 THEN
0062          IF high=2 THEN (* long branch *)
007E            opcode_type=7
0085            mnemonic$='lb'+MID$('rnhilscccsneeqvcvsplmigeltgtle',
                2*low+1,2)
00BB          ELSE
00BF            IF opcode=$3F THEN
00CC              opcode_type=11 ¥(* os9 *)
00DC              mnemonic$='os9'
00E6            ELSE (* undefined *)
00F8              opcode_type=0
00FF            ENDIF
0101          ENDIF
0103        ELSE (* cmpd,cmpy,ldy,lds,sty,sts *)
0125          opcode_type=MOD(high-8,4)+2
0136          IF opcode_type=2 THEN
0142            opcode_type=1 ¥(* word immidiate *)
015D          ENDIF
015F          IF high<=$0B THEN (* cmpd,cmpy,ldy,sty *)
0182            mnemonic$=TRIM$(MID$('********cmpd****************************
***cmpy****ldy sty '
                ,4*low+1,4))
01D2          ELSE (* lds,sts *)
01E2            mnemonic$=MID$('*************************************ldssts'
                ,3*low+1,3)
0222          ENDIF
0224          IF opcode=$8F OR opcode=$CF OR LEFT$(mnemonic$,1)='*' THEN
               (* undefined *)
0252            opcode_type=0
0259          ENDIF
025B        ENDIF
025D      ELSE (* low=0 *)
026B        opcode_type=0
0272      ENDIF
0274      END
PROCEDURE opcode_pre_11
0000      PARAM opcode:BYTE; opcode_type:INTEGER; mnemonic$:STRING[5]
```

```
0018      DIM high,low:BYTE
0023      IF opcode=$3F THEN
0030        opcode_type=10 ¥(* inherent *)
0045        mnemonic$='swi3'
0050      ELSE
0054        high=opcode/$10
0060        low=opcode-high*$10
0070        opcode_type=high-6
007B        IF opcode_type=2 THEN
0087          opcode_type=1
008E        ENDIF
0090        IF low=3 THEN (* cmpu *)
00A5          mnemonic$='cmpu'
00B0        ELSE
00B4          IF low=$0C THEN (* cmps *)
00CA            mnemonic$='cmps'
00D5          ELSE (* undefined *)
00E7            opcode_type=0
00EE          ENDIF
00F0        ENDIF
00F2      ENDIF
00F4      END
PROCEDURE word_immediate
0000      PARAM object_path_no,label_fp,address,m_code_ptr:INTEGER; m_code
          (20):BYTE; end_flag:BOOLEAN; operand$,comment$:STRING
         [50]
0033      DIM imd_value:INTEGER; hex_str$:STRING[4]; label$:STRING[8]
          ; resist:BOOLEAN
0056      RUN get_next_word(object_path_no,address,imd_value,end_flag
          )
006F      IF end_flag THEN
0078        END
007A      ENDIF
007C      RUN label_manager(label_fp,imd_value,$20,label$,resist)
0099      IF resist THEN
00A2        operand$='#'+label$
00AE      ELSE
00B2        RUN make_4hex(imd_value,hex_str$)
00C1        operand$='#$'+hex_str$
00CE      ENDIF
00D0      m_code(m_code_ptr+1)=PEEK(ADDR(imd_value))
00E2      m_code(m_code_ptr+2)=PEEK(ADDR(imd_value)+1)
00F7      m_code_ptr=m_code_ptr+2
0102      END
PROCEDURE byte_immediate
0000      PARAM object_path_no,address,m_code_ptr:INTEGER; m_code(20)
          :BYTE; end_flag:BOOLEAN; operand$,comment$:STRING[50]

002F      DIM imd_value:BYTE; hex_str$:STRING[2]
0041      RUN get_next_byte(object_path_no,address,imd_value,end_flag
          )
005A      IF end_flag THEN
0063        END
0065      ENDIF
0067      RUN make_2hex(imd_value,hex_str$)
0076      operand$='#$'+hex_str$
0083      IF LAND(imd_value,$7F))=$20 THEN (* ascii character *)
00A8        comment$=comment$+' '+CHR$(LAND(imd_value,$7F))+''
00C2      ENDIF
00C4      m_code(m_code_ptr+1)=imd_value
00D3      m_code_ptr=m_code_ptr+1
00DE      END
PROCEDURE direct_page
0000      PARAM object_path_no,label_fp,address,m_code_ptr:INTEGER; m_code
          (20):BYTE; end_flag:BOOLEAN; operand$,comment$:STRING
         [50]
0033      DIM dir_value:BYTE; hex_str$:STRING[2]; label$:STRING[8]; resist
          :BOOLEAN
0056      RUN get_next_byte(object_path_no,address,dir_value,end_flag
          )
006F      IF end_flag THEN
0078        END
007A      ENDIF
007C      IF m_code(1)=$0E OR m_code(1)=$9D THEN
0095        RUN label_manager(label_fp,dir_value+0,-($40),label$,resist
            )
00B5      ELSE (* data *)
00C2        RUN label_manager(label_fp,dir_value+0,-($80),label$,resist
            )
00E2      ENDIF
00E4      IF resist THEN
00ED        operand$='<'+label$
00F9      ELSE
00FD        RUN make_2hex(dir_value,hex_str$)
010C        operand$='<$'+hex_str$
0119      ENDIF
011B      m_code(m_code_ptr+1)=dir_value
012A      m_code_ptr=m_code_ptr+1
```

```
 0135       END
PROCEDURE extended
 0000       PARAM object_path_no,label_fp,address,m_code_ptr:INTEGER; m_code
             (20):BYTE; end_flag:BOOLEAN; operand$,comment$:STRING
             [50]
 0033       DIM ext_value:INTEGER; hex_str$:STRING[4]; label$:STRING[8]
             ; resist:BOOLEAN
 0056       RUN get_next_word(object_path_no,address,ext_value,end_flag
             )
 006F       IF end_flag THEN
 0078         END
 007A       ENDIF
 007C       IF m_code(1)=$7E OR m_code(1)=$BD THEN (* jmp(EXT),jsr(EXT) *)
 00AB         RUN label_manager(label_fp,ext_value,$10,label$,resist)
 00C8       ELSE
 00CC         RUN label_manager(label_fp,ext_value,$20,label$,resist)
 00E9       ENDIF
 00EB       IF resist THEN
 00F4         operand$='>'+label$
 0100       ELSE
 0104         RUN make_4hex(ext_value,hex_str$)
 0113         operand$='>$'+hex_str$
 0120       ENDIF
 0122       m_code(m_code_ptr+1)=PEEK(ADDR(ext_value))
 0134       m_code(m_code_ptr+2)=PEEK(ADDR(ext_value)+1)
 0149       m_code_ptr=m_code_ptr+2
 0154       END
PROCEDURE short_branch
 0000       PARAM object_path_no,label_fp,address,m_code_ptr:INTEGER; m_code
             (20):BYTE; end_flag:BOOLEAN; operand$,comment$:STRING
             [50]
 0033       DIM rel_value:BYTE; hex_str$:STRING[4]; label$:STRING[8]; resist
             :BOOLEAN; absolute_address,offset_value:INTEGER
 0060       RUN get_next_byte(object_path_no,address,rel_value,end_flag
             )
 0079       IF end_flag THEN
 0082         END
 0084       ENDIF
 0086       RUN byte_to_int(rel_value,offset_value)
 0095       absolute_address=address+offset_value
 00A1       RUN label_manager(label_fp,absolute_address,$04,label$,resist
             )
 00BE       IF resist THEN
 00C7         operand$=label$
 00CF       ELSE
 00D3         RUN make_4hex(absolute_address,hex_str$)
 00E2         operand$='$'+hex_str$
 00EE       ENDIF
 00F0       m_code(m_code_ptr+1)=rel_value
 00FF       m_code_ptr=m_code_ptr+1
 010A       END
PROCEDURE long_branch
 0000       PARAM object_path_no,label_fp,address,m_code_ptr:INTEGER; m_code
             (20):BYTE; end_flag:BOOLEAN; operand$,comment$:STRING
             [50]
 0033       DIM rel_value:INTEGER; hex_str$:STRING[4]; label$:STRING[8]
             ; resist:BOOLEAN; absolute_address:INTEGER
 005C       RUN get_next_word(object_path_no,address,rel_value,end_flag
             )
 0075       IF end_flag THEN
 007E         END
 0080       ENDIF
 0082       absolute_address=address+rel_value
 008E       RUN label_manager(label_fp,absolute_address,$04,label$,resist
             )
 00AB       IF resist THEN
 00B4         operand$=label$
 00BC       ELSE
 00C0         RUN make_4hex(absolute_address,hex_str$)
 00CF         operand$='$'+hex_str$
 00DB       ENDIF
 00DD       m_code(m_code_ptr+1)=PEEK(ADDR(rel_value))
 00EF       m_code(m_code_ptr+2)=PEEK(ADDR(rel_value)+1)
 0104       m_code_ptr=m_code_ptr+2
 010F       END
PROCEDURE psh_pul
 0000       PARAM object_path_no,address:INTEGER; term:BOOLEAN; m_code_ptr
             :INTEGER; m_code(20):BYTE; end_flag:BOOLEAN; operand$
             ,comment$:STRING[50]; reg$:STRING[1]; illegal:BOOLEAN
 0048       DIM post_byte,pb:BYTE
 0053       illegal=FALSE
 0059       RUN get_next_byte(object_path_no,address,post_byte,end_flag
             )
 0072       IF end_flag THEN
 007B         END
 007D       ENDIF
 007F       pb=post_byte
 0087       FOR bit=0 TO 7
 0099         IF MOD(pb,2)=1 THEN
 00A8           operand$=operand$+TRIM$(MID$('cca b dpx y '+reg$+' pc',
                 bit*2+1,2))+','
 00DD           IF bit=7 THEN
 00EA             term=TRUE
 00F0           ENDIF
 00F2         ENDIF
 00F4         pb=pb/2
 00FF       NEXT bit
 010A       IF RIGHT$(operand$,1)=',' THEN
 011A         operand$=MID$(operand$,1,LEN(operand$)-1)
 012C       ELSE (* illegal *)
 013C         illegal=TRUE
 0142       ENDIF
 0144       IF m_code(m_code_ptr)=$34 OR m_code(m_code_ptr)=$36 THEN
 015F         term=FALSE
 0165       ENDIF
 0167       m_code(m_code_ptr+1)=post_byte
 0176       m_code_ptr=m_code_ptr+1
 0181       END
PROCEDURE tfr_exg
 0000       PARAM object_path_no,address:INTEGER; term:BOOLEAN; m_code_ptr
             :INTEGER; m_code(20):BYTE; end_flag:BOOLEAN; operand$
             ,comment$:STRING[50]; illegal:BOOLEAN
 003D       DIM post_byte,source,destination:BYTE
 004C       RUN get_next_byte(object_path_no,address,post_byte,end_flag
             )
 0065       IF end_flag THEN
 006E         END
 0070       ENDIF
 0072       source=post_byte/$10
 007E       destination=post_byte-source*$10
 008E       operand$=TRIM$(MID$('d x y u s pc****a b ccdp',source*2+1,2
             ))+','+TRIM$(MID$('d x y u s pc****a b ccdp',destination
             *2+1,2))
 00E6       IF m_code(m_code_ptr)=$1E AND (source=5 OR destination=5) THEN

 0104         term=TRUE
 010A       ELSE
 010E         IF m_code(m_code_ptr)=$1F AND destination=5 THEN
 0125           term=TRUE
 012B         ENDIF
 012D       ENDIF
 012F       m_code(m_code_ptr+1)=post_byte
 013E       m_code_ptr=m_code_ptr+1
 0149       IF source<$06 AND destination<$06 THEN
 015E         illegal=FALSE
 0164       ELSE
 0168         IF source>=$08 AND destination>=$08 AND source<=$0B AND destination
               <=$0B THEN
 018D           illegal=FALSE
 0193         ELSE
 0197           illegal=TRUE
 019D         ENDIF
 019F       ENDIF
 01A1       END
PROCEDURE indexed
 0000       PARAM object_path_no,label_fp,address,m_code_ptr:INTEGER; m_code
             (20):BYTE; end_flag:BOOLEAN; operand$,comment$:STRING
             [50]; illegal:BOOLEAN
 0039       DIM post_byte,byte_offset,bit_7,bit_06,reg,bit_04,mode,kind
             :BYTE; word_offset,absolute_address:INTEGER; register$
             :STRING[1]; label$:STRING[8]; hex_str$:STRING[4]; resist
             :BOOLEAN
 008D       DIM plus_byte:BYTE
 0094       illegal=FALSE
 009A       RUN get_next_byte(object_path_no,address,post_byte,end_flag
             )
 00B3       IF end_flag THEN
 00BC         END
 00BE       ENDIF
 00C0       m_code(m_code_ptr+1)=post_byte
 00CF       m_code_ptr=m_code_ptr+1
 00DA       IF post_byte=$9F THEN (* extended indirect *)
 00FD         RUN extended(object_path_no,label_fp,address,m_code_ptr,m_code
               ,end_flag,operand$,comment$)
 012A         operand$='['+RIGHT$(operand$,LEN(operand$)-1)+']'
 0142       ELSE (* others *)
 0151         bit_7=post_byte/$80
 015D         bit_06=post_byte-bit_7*$80
 016D         reg=bit_06/$20
 0179         bit_04=bit_06-reg*$20
 0189         mode=bit_04/$10
 0195         kind=bit_04-mode*$10
 01A5         register$=MID$('xyus',reg+1,1)
 01B9         IF bit_7=1 AND kind=$04 THEN (* nooffset *)
 01DA           operand$=','+register$
 01E6           plus_byte=0
```

```
01ED         ELSE
01F1           IF bit_7=0 THEN (* 5 bit offset *)
020E             IF bit_04>=16 THEN
021A               word_offset=bit_04-32
0225             ELSE
0229               word_offset=bit_04
0231             ENDIF
0233             operand$=STR$(word_offset)+','+register$
0244             plus_byte=0
024B           ELSE
024F             IF bit_7=1 AND kind=$08 THEN (* 8 bit offset *)
0274               RUN get_next_byte(object_path_no,address,byte_offset
                    ,end_flag)
028D               IF end_flag THEN
0296                 END
0298               ENDIF
029A               RUN byte_to_int(byte_offset,word_offset)
02A9               operand$=STR$(word_offset)+','+register$
02BA               plus_byte=1
02C1             ELSE
02C5               IF bit_7=1 AND kind=$09 THEN (* 16 bit offset *)
02EB                 RUN get_next_word(object_path_no,address,word_offset
                      ,end_flag)
0304                 IF end_flag THEN
030D                   END
030F                 ENDIF
0311                 operand$=STR$(word_offset)+','+register$
0322                 plus_byte=2
0329               ELSE
032D                 IF bit_7=1 AND mode=0 AND kind=$00 THEN (* auto increment +
1 *)
035E                   operand$=','+register$+'+'
036E                   plus_byte=0
0375                 ELSE
0379                   IF bit_7=1 AND kind=$01 THEN (* auto increment +2 )
03A2                     operand$=','+register$+'++'
03B3                     plus_byte=0
03BA                   ELSE
03BE                     IF bit_7=1 AND mode=0 AND kind=$02 THEN (* auto decreme
nt -1 *)
03EF                       operand$=',-'+register$
03FC                       plus_byte=0
0403                     ELSE
0407                       IF bit_7=1 AND kind=$03 THEN (* auto decrement -2 *)
0431                         operand$=',--'+register$
043F                         plus_byte=0
0446                       ELSE
044A                         IF bit_7=1 AND kind=$0C THEN (* pc_rel 8 bit offset
*)
0476                           RUN get_next_byte(object_path_no,address
                                ,byte_offset,end_flag)
048F                           IF end_flag THEN
0498                             END
049A                           ENDIF
049C                           RUN byte_to_int(byte_offset,word_offset
                                )
04AB                           absolute_address=address+word_offset
04B7                           IF m_code(1)=$6E OR m_code(1)=$AD THEN (*
                                jmp(IND),jsr(IND) *)
04E6                             RUN label_manager(label_fp,absolute_address
                                  ,$04,label$,resist)
0503                           ELSE (* data *)
0510                             RUN label_manager(label_fp,absolute_address
                                  ,$08,label$,resist)
052D                           ENDIF
052F                           IF resist THEN
0538                             operand$=label$+',pcr'
0547                           ELSE
054B                             RUN make_4hex(absolute_address,hex_str$
                                  )
055A                             operand$='$'+hex_str$+',pcr'
056D                           ENDIF
056F                           plus_byte=1
0576                         ELSE
057A                           IF bit_7=1 AND kind=$0D THEN (* pc_rel 16 bit off
set *)
05A7                             RUN get_next_word(object_path_no,address
                                  ,word_offset,end_flag)
05C0                             IF end_flag THEN
05C9                               END
05CB                             ENDIF
05CD                             absolute_address=address+word_offset
05D9                             IF m_code(1)=$6E OR m_code(1)=$AD THEN
                                  (* jmp(IND),jsr(IND) *)
0608                               RUN label_manager(label_fp,absolute_address
                                    ,$04,label$,resist)
0625                             ELSE (* data *)
0632                               RUN label_manager(label_fp,absolute_address
                                    ,$08,label$,resist)
064F                             ENDIF
0651                             IF resist THEN
065A                               operand$=label$+',pcr'
0669                             ELSE
066D                               RUN make_4hex(absolute_address,hex_str$
                                    )
067C                               operand$='$'+hex_str$+',pcr'
068F                             ENDIF
0691                             plus_byte=2
0698                           ELSE
069C                             IF bit_7=1 AND kind=$06 THEN (* a-register offs
et *)
06C6                               operand$='a,'+register$
06D3                               plus_byte=0
06DA                             ELSE
06DE                               IF bit_7=1 AND kind=$05 THEN (* b-register of
fset *)
0708                                 operand$='b,'+register$
0715                                 plus_byte=0
071C                               ELSE
0720                                 IF bit_7=1 AND kind=$0B THEN (* d-register
offset *)
074A                                   operand$='d,'+register$
0757                                   plus_byte=0
075E                                 ELSE (* undefined *)
0770                                   plus_byte=0
0777                                   illegal=TRUE
077D                                   (* ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆ *)
0799                                 ENDIF
079B                               ENDIF
079D                             ENDIF
079F                           ENDIF
07A1                         ENDIF
07A3                       ENDIF
07A5                     ENDIF
07A7                   ENDIF
07A9                 ENDIF
07AB               ENDIF
07AD             ENDIF
07AF           ENDIF
07B1         ENDIF
07B3         IF mode=1 AND bit_7=1 THEN (* indirect *)
07D3           operand$='['+operand$+']'
07E3         ENDIF
07E5         IF plus_byte>0 THEN
07F1           IF plus_byte=1 THEN
07FD             m_code(m_code_ptr+1)=byte_offset
080C             m_code_ptr=m_code_ptr+1
0817           ELSE (* plus_byte=2 *)
082B             m_code(m_code_ptr+1)=PEEK(ADDR(word_offset))
083D             m_code(m_code_ptr+2)=PEEK(ADDR(word_offset)+1)
0852             m_code_ptr=m_code_ptr+2
085D           ENDIF
085F         ENDIF
0861       ENDIF
0863       END
PROCEDURE make_2hex
0000       PARAM a:BYTE; hex$:STRING[2]
0012       DIM b,nibble:BYTE
001D       hex$=''
0024       b=a
002C       FOR i=0 TO 1
003E         nibble=b/$10
004A         b=MOD(b,$10)*$10
005A         hex$=hex$+MID$('0123456789abcdef',nibble+1,1)
007E       NEXT i
0089       END
PROCEDURE make_4hex
0000       PARAM a:INTEGER; hex$:STRING[4]
0012       DIM h$:STRING[2]; byt:BYTE
0024       hex$=''
002B       FOR i=0 TO 1
003D         byt=PEEK(ADDR(a)+i)
004E         RUN make_2hex(byt,h$)
005D         hex$=hex$+h$
0069       NEXT i
0074       END
PROCEDURE get_next_byte
0000       PARAM object_path_no,address:INTEGER; value:BYTE; end_flag:
            BOOLEAN
0017       DIM real_adr:REAL
001E       RUN int_to_real(address,real_adr)
002D       SEEK #object_path_no,real_adr
0037       end_flag=EOF(#object_path_no)
0040       IF end_flag=FALSE THEN
004B         GET #object_path_no,value
0055         address=address+1
```

```
 0060       ENDIF
 0062       END
PROCEDURE get_next_word
 0000       PARAM object_path_no,address,value:INTEGER; end_flag:BOOLEAN
 0015       DIM h_val,l_val:BYTE
 0020       RUN get_next_byte(object_path_no,address,h_val,end_flag)
 0039       IF end_flag=TRUE THEN
 0044         END
 0046       ELSE
 004A         RUN get_next_byte(object_path_no,address,l_val,end_flag)
 0063         POKE ADDR(value),h_val
 006E         POKE ADDR(value)+1,l_val
 007C       ENDIF
 007E       END
PROCEDURE os9
 0000       PARAM object_path_no,address:INTEGER; term:BOOLEAN; m_code_ptr
             :INTEGER; m_code(20):BYTE; end_flag:BOOLEAN; operand$
             ,comment$:STRING[50]
 0037       DIM sys$:STRING[256]; post_byte:BYTE
 0049       RUN get_next_byte(object_path_no,address,post_byte,end_flag
             )
 0062       IF post_byte<$1F THEN (* user mode function request *)
 008E         operand$=TRIM$('F$'+MID$('Link  Load  UnLinkFork  Wait  Chain Exit
 Mem   Send  Icpt  Sleep SSpd  ID    SPriorSSWI  PErr  PrsNamCmpNamSchBitAllBitD
elBitTime  STime CRC   GPrDscGBlkMpGModDrCpyMemSUser UnLoadKanji '
              ,post_byte*6+1,6))
 0161       ELSE
 0165         IF post_byte>=$28 AND post_byte<=$4E THEN (* system mode function r
equest *)
 019B           sys$='SRqMemSRtMemIRQ   IOQu  AProc NProc VModulFind64All64 Ret64
 SSvc  IODel SLink Boot  BtMem GProcPMove  AllRAMAllImgDelImgSetImgFreeLBFreeHBA
llTskDelTskSetTskResTskRelTskDATLogDATTmpLDAXY LDAXYPLDDDXYLDABX STABX AllPrcDel
PrcELink FModul'
 028C           operand$=TRIM$('F$'+MID$(sys$,(post_byte-$28)*6+1,6))
 02AA         ELSE
 02AE           IF post_byte>=$80 AND post_byte<=$90 THEN (* I/O service request
*)
 02DB             operand$=TRIM$('I$'+MID$('AttachDetachDup   CreateOpen  MakDirC
hgDirDeleteSeek  Read  Write ReadLnWritLnGetSttSetSttClose DeletX'
                ,(post_byte-$80)*6+1,6))
 035E           ELSE (* undefined service request *)
 0380           ENDIF
 0382         ENDIF
 0384       ENDIF
 0386       IF post_byte=$05 OR post_byte=$06 OR post_byte=$04 OR post_byte
             =$0A OR post_byte=$2D THEN
 03B3         term=TRUE
 03B9       ENDIF
 03BB       m_code(m_code_ptr+1)=post_byte
 03CA       m_code_ptr=m_code_ptr+1
 03D5       END
PROCEDURE module_header
 0000       PARAM object_path_no,label_fp,address,m_code_ptr:INTEGER; m_code
             (20):BYTE; end_flag:BOOLEAN; operand$,comment$:STRING
             [50]
 0033       PARAM module_size,module_name_offset,execute_offset,parmanent_storage
_size
             :INTEGER; type_language,attr_rev,header_parity:BYTE
 0054       PARAM not_6809:BOOLEAN
 005B       PARAM emod_adr:INTEGER
 0062       DIM label$:STRING[8]; resist:BOOLEAN
 0074       DIM hex_str2$:STRING[2]; hex_str4$:STRING[4]
 008B       RUN get_next_word(object_path_no,address,module_size,end_flag
             )
 00A4       IF end_flag THEN
 00AD         END
 00AF       ENDIF
 00B1       RUN get_next_word(object_path_no,address,module_name_offset
             ,end_flag)
 00CA       IF end_flag THEN
 00D3         END
 00D5       ENDIF
 00D7       RUN get_next_byte(object_path_no,address,type_language,end_flag
             )
 00F0       IF end_flag THEN
 00F9         END
 00FB       ENDIF
 00FD       RUN get_next_byte(object_path_no,address,attr_rev,end_flag)

 0116       IF end_flag THEN
 011F         END
 0121       ENDIF
 0123       RUN get_next_byte(object_path_no,address,header_parity,end_flag
             )
 013C       IF end_flag THEN
 0145         END
 0147       ENDIF
 0149       RUN get_next_word(object_path_no,address,execute_offset,end_flag
             )
 0162       IF end_flag THEN
 016B         END
 016D       ENDIF
 016F       RUN get_next_word(object_path_no,address,parmanent_storage_size
             ,end_flag)
 0188       IF end_flag THEN
 0191         END
 0193       ENDIF
 0195       PRINT '***************************'
 01B4       PRINT '     module type= '; MID$('any progsubrmultdata****************
************sys fmanddrvddsc'
             ,4*INT(type_language/$10)+1,4)
 0220       PRINT '     language    = '; MID$('any 6809IcodPcodCobl',4*MOD
             (type_language,$10)+1,4)
 025C       RUN make_2hex(attr_rev,hex_str2$)
 026B       PRINT '     attribute   = '; MID$('non re-entre-entrant',10*INT
             (attr_rev/$80)+1,10)
 02AB       PRINT '     revision    = $'; RIGHT$(hex_str2$,1)
 02C8       PRINT '***************************'
 02E7       emod_adr=module_size-3
 02F2       RUN label_manager(label_fp,module_size-3,$04,label$,resist)

 0311       IF resist THEN
 031A         operand$=label$+'+3,'
 0328       ELSE
 032C         RUN make_4hex(module_size,hex_str4$)
 033B         operand$=operand$+'$'+hex_str4$+','
 034F       ENDIF
 0351       RUN label_manager(label_fp,module_name_offset,$0B,label$,resist
             )
 036E       IF resist THEN
 0377         operand$=operand$+label$+','
 0387       ELSE
 038B         RUN make_4hex(module_name_offset,hex_str4$)
 039A         operand$=operand$+'$'+hex_str4$+','
 03AE       ENDIF
 03B0       RUN make_2hex(type_language,hex_str2$)
 03BF       operand$=operand$+'$'+hex_str2$+','
 03D3       RUN make_2hex(attr_rev,hex_str2$)
 03E2       operand$=operand$+'$'+hex_str2$+','
 03F6       RUN label_manager(label_fp,execute_offset,$04,label$,resist
             )
 0413       IF resist THEN
 041C         operand$=operand$+label$+','
 042C       ELSE
 0430         RUN make_4hex(execute_offset,hex_str4$)
 043F         operand$=operand$+'$'+hex_str4$+','
 0453       ENDIF
 0455       RUN make_4hex(parmanent_storage_size,hex_str4$)
 0464       operand$=operand$+'$'+hex_str4$
 0474       m_code(m_code_ptr+1)=PEEK(ADDR(module_size))
 0486       m_code(m_code_ptr+2)=PEEK(ADDR(module_size)+1)
 049B       m_code(m_code_ptr+3)=PEEK(ADDR(module_name_offset))
 04AD       m_code(m_code_ptr+4)=PEEK(ADDR(module_name_offset)+1)
 04C2       m_code(m_code_ptr+5)=type_language
 04D1       m_code(m_code_ptr+6)=attr_rev
 04E0       m_code(m_code_ptr+7)=header_parity
 04EF       m_code(m_code_ptr+8)=PEEK(ADDR(execute_offset))
 0501       m_code(m_code_ptr+9)=PEEK(ADDR(execute_offset)+1)
 0516       m_code(m_code_ptr+10)=PEEK(ADDR(parmanent_storage_size))
 0528       m_code(m_code_ptr+11)=PEEK(ADDR(parmanent_storage_size)+1)
 053D       m_code_ptr=m_code_ptr+11
 0548       IF MOD(type_language,$10)=1 OR MOD(type_language,$10)=0 THEN

 0563         not_6809=FALSE
 0569       ELSE (* not 6809 code *)
 057F         not_6809=TRUE
 0585       ENDIF
 0587       END
PROCEDURE byte_to_int
 0000       PARAM a:BYTE; b:INTEGER
 000D       IF a<128 THEN
 0019         b=a
 0021       ELSE (* a>=128 *)
 0030         b=a-256
 003C       ENDIF
 003E       END
PROCEDURE int_to_real
 0000       PARAM int_value:INTEGER; real_value:REAL
 000D       IF int_value>=0 THEN
 0019         real_value=int_value
 0022       ELSE (* minus *)
 0030         real_value=FLOAT(int_value)+65536.
 0040       ENDIF
 0042       END
PROCEDURE label_initialize
 0000       PARAM filename$:STRING[32]; fp:INTEGER
```

```
0012       PARAM init_mode:INTEGER
0019       DIM i:INTEGER
0020       DIM inidata(256):BYTE
002C       IF init_mode=1 THEN (* create new file *)
004C         CREATE #fp,filename$
0056         SEEK #fp,65535.
0063       ELSE
0067         OPEN #fp,filename$:UPDATE
0073       ENDIF
0075       SEEK #fp,.0 ¥(* restore *)
008F       IF init_mode<>2 THEN
009B         PRINT 'Initializing label file ,wait a moment!'
00C6         FOR i=1 TO 256
00D7           inidata(i)=0
00E2         NEXT i
00ED         FOR i=0 TO 255
00FD           PUT #fp,inidata
0107         NEXT i
0112       ENDIF
0114       END
PROCEDURE label_manager
0000       PARAM label_fp:INTEGER
0007       PARAM ref_address,ref_type:INTEGER; label$:STRING[8]; resist
            :BOOLEAN
0023       DIM addr_real:REAL
002A       DIM hex2$:STRING[2]; hex4$:STRING[4]
0041       DIM new_type:BYTE
0048       DIM old_type:BYTE
004F       IF label_fp=0 THEN
005B         resist=FALSE
0061       ELSE
0065         IF label_fp<>1 THEN
0071           RUN int_to_real(ref_address,addr_real)
0080           SEEK #label_fp,addr_real
008A           GET #label_fp,old_type
0094           SEEK #label_fp,addr_real
009E         ENDIF
00A0         IF ref_type>=0 THEN (* not direct *)
00BB           IF label_fp<>1 THEN
00C7             new_type=LOR(LAND(LOR(old_type,ref_type),$FC),LAND(ref_type
                  ,$03))
00DF             PUT #label_fp,new_type
00E9           ENDIF
00EB           RUN make_4hex(ref_address,hex4$)
00FA           label$=MID$('??IDIPEDEPPDPP',2*SUBSTR(CHR$(LAND(ref_type
                ,$FC)),CHR$($80)+CHR$($40)+CHR$($20)+CHR$($10)+CHR$
                ($08)+CHR$($04))+1,2)+'_'+hex4$
0146         ELSE (* direct label *)
015B           IF label_fp<>1 THEN
0167             new_type=LOR(old_type,ref_type)
0173             PUT #label_fp,new_type
017D           ENDIF
017F           RUN make_4hex(ref_address,hex4$)
018E           label$=MID$('??DDDP',2*SUBSTR(CHR$(LAND(-(ref_type),$C0
                )),CHR$($80)+CHR$($40))+1,2)+'_'+RIGHT$(hex4$,2)
01C2         ENDIF
01C4         resist=TRUE
01CA       ENDIF
01CC       END
PROCEDURE label_reference
0000       PARAM label_fp:INTEGER
0007       PARAM label_address:INTEGER; label$:STRING[8]; find:BOOLEAN
001F       PARAM l_attr:BYTE
0026       PARAM e_attr:BYTE
002D       PARAM max_address:INTEGER
0034       DIM addr_real:REAL
003B       DIM hex4$:STRING[4]
0047       DIM ref_type:BYTE
004E       DIM n:INTEGER
0055       DIM real_max_address:REAL
005C       IF max_address=$FFFF THEN (* first access... ignore *)
0084         find=FALSE
008A       ELSE
008E         IF l_attr=0 THEN (* next address *)
00AB           label_address=1+label_address
00B6         ENDIF
00B8         RUN int_to_real(label_address,addr_real)
00C7         RUN int_to_real(max_address,real_max_address)
00D6         find=FALSE
00DC         WHILE find=FALSE AND addr_real<real_max_address+1 DO
00F3           SEEK #label_fp,addr_real
00FD           GET #label_fp,ref_type
0107           IF l_attr=0 THEN
0113             l_attr=LAND(ref_type,e_attr)
011F           ENDIF
0121           IF LAND(LAND(ref_type,$FC),LAND(e_attr,l_attr))<>0 THEN
                (* found *)
0143             find=TRUE
0149             RUN make_4hex(label_address,hex4$)
0158             RUN find_attr(l_attr,n)
0167             IF n=1 OR n=2 THEN (* direct mode *)
018B               hex4$=RIGHT$(hex4$,2)
0196             ENDIF
0198             label$=MID$('DDDPEDEPPDPP',2*n-1,2)+'_'+hex4$
01BF           ENDIF
01C1           addr_real=1.+addr_real
01D0           label_address=1+label_address
01DB         ENDWHILE
01DF         label_address=label_address-1
01EA       ENDIF
01EC       END
PROCEDURE disasm_1_line
0000       (* ♥♥♥♥ disassemble 1 line ♥♥♥♥
001F       PARAM object_path_no:INTEGER
0026       PARAM label_fp:INTEGER
002D       PARAM address:INTEGER; term:BOOLEAN
003A       PARAM word_opcode:BOOLEAN; code_count:INTEGER; machine_code
            (20):BYTE; opcode_type:INTEGER; end_flag:BOOLEAN; mnemonic$
            :STRING[5]; operand$:STRING[50]; comment$:STRING[50]
007F       PARAM mod_siz,mod_nam,exec_ofs,st_siz:INTEGER; typ_lng,atr_rev
            ,head_par:BYTE; not_6809:BOOLEAN
00A6       PARAM emod_adr:INTEGER
00AD       DIM machine_code_pointer:INTEGER
00B4       DIM illegal:BOOLEAN
00BB       DIM hex2$:STRING[2]
00C7       mnemonic$=''
00CE       opcode$=''
00D5       comment$=''
00DC       term=FALSE
00E2       RUN opcode_main(object_path_no,address,term,opcode_type,end_flag
            ,mnemonic$,word_opcode,machine_code,emod_adr)
0114       IF end_flag THEN
011D         END
011F       ENDIF
0121       IF word_opcode=TRUE THEN
012C         machine_code_pointer=2
0133       ELSE
0137         machine_code_pointer=1
013E       ENDIF
0140       IF opcode_type=0 THEN (* undefined *)
015A       ELSE
015E         IF opcode_type=1 THEN (* word immediate *)
017E           RUN word_immediate(object_path_no,label_fp,address,machine_code_p
ointer
            ,machine_code,end_flag,operand$,comment$)
01AB         ELSE
01AF           IF opcode_type=2 THEN (* byte immediate *)
01CE             RUN byte_immediate(object_path_no,address,machine_code_pointer
                  ,machine_code,end_flag,operand$,comment$)
01F6           ELSE
01FA             IF opcode_type=3 THEN (* direct page *)
0216               RUN direct_page(object_path_no,label_fp,address,machine_code_
pointer
                ,machine_code,end_flag,operand$,comment$)
0243             ELSE
0247               IF opcode_type=4 THEN (* indexed *)
025F                 RUN indexed(object_path_no,label_fp,address,machine_code_po
inter
                  ,machine_code,end_flag,operand$,comment$,illegal
                  )
0291                 IF illegal=TRUE THEN
029C                   opcode_type=0
02A3                 ENDIF
02A5               ELSE
02A9                 IF opcode_type=5 THEN (* extended *)
02C2                   RUN extended(object_path_no,label_fp,address,machine_code
_pointer
                    ,machine_code,end_flag,operand$,comment$)

02EF                 ELSE
02F3                   IF opcode_type=6 THEN (* short_branch *)
0310                     RUN short_branch(object_path_no,label_fp,address
                          ,machine_code_pointer,machine_code,end_flag
                          ,operand$,comment$)
033D                   ELSE
0341                     IF opcode_type=7 THEN (* long_branch *)
035D                       RUN long_branch(object_path_no,label_fp,address
                            ,machine_code_pointer,machine_code,end_flag
                            ,operand$,comment$)
038A                     ELSE
038E                       IF opcode_type=8 THEN (* push pull *)
03A8                         IF machine_code(1)=$34 OR machine_code(1)
                              =$35 THEN (* pshs,puls *)
03CF                           RUN psh_pul(object_path_no,address,term
                                ,machine_code_pointer,machine_code
                                ,end_flag,operand$,comment$,'u',illegal
```

```
0405                ELSE (* pshu,pulu *)
0417                  RUN psh_pul(object_path_no,address,term
                       ,machine_code_pointer,machine_code
                       ,end_flag,operand$,comment$,'s',illegal
                       )
044D                ENDIF
044F                IF illegal=TRUE THEN
045A                  opcode_type=0
0461                ENDIF
0463              ELSE
0467                IF opcode_type=9 THEN (* tfr exg *)
047F                  RUN tfr_exg(object_path_no,address,term
                       ,machine_code_pointer,machine_code
                       ,end_flag,operand$,comment$,illegal
                       )
04B1                  IF illegal=TRUE THEN
04BC                    opcode_type=0
04C3                  ENDIF
04C5                ELSE
04C9                  IF opcode_type=10 THEN (* inherent *)
04E2                    (* do none *)
04EF                  ELSE
04F3                    IF opcode_type=11 THEN (* os9 *)
0507                      RUN os9(object_path_no,address,term
                           ,machine_code_pointer,machine_code
                           ,end_flag,operand$,comment$)
0534                    ELSE
0538                      IF opcode_type=-1 THEN (* sync_byte *)
0553                        RUN module_header(object_path_no,
                             label_fp,address,machine_code_pointe
                             ,machine_code,end_flag,operand$
                             ,comment$,mod_siz,mod_nam,exec_ofs
                             ,st_size,typ_lng,atr_rev,head_par
                             ,not_6809,emod_adr)
05AD                      ELSE
05B1                        IF opcode_type=-2 THEN (* emod *)
05C7                          RUN emod(object_path_no,address
                               ,term,machine_code_pointer
                               ,machine_code,end_flag,operand$
                               ,comment$)
05F4                        ELSE
05F8                          ERROR 1 ¥(* system error *)
060E                        ENDIF
0610                      ENDIF
0612                    ENDIF
0614                  ENDIF
0616                ENDIF
0618              ENDIF
061A            ENDIF
061C          ENDIF
061E        ENDIF
0620      ENDIF
0622    ENDIF
0624  ENDIF
0626  ENDIF
0628  ENDIF
062A  IF opcode_type=0 THEN (* undefined instruction *)
0651    mnemonic$='FCB'
065B    RUN make_2hex(machine_code(1),hex2$)
066C    operand$='$'+hex2$
0678    IF LAND(machine_code(1),$7F)>=$20 THEN (* ascii char. *)
069B      comment$=comment$+' '+CHR$(LAND(machine_code(1),$7F))+
          ''''
06B7    ENDIF
06B9    code_count=1
06C0    address=address-machine_code_pointer+1
06CF  ELSE
06D3    code_count=machine_code_pointer
06DB  ENDIF
06DD  END
PROCEDURE make_mcasc
0000  PARAM m_code(20):BYTE
000C  PARAM c_cnt:INTEGER
0013  PARAM hex$:STRING[50]
001F  PARAM ascii$:STRING[20]
002B  DIM h$:STRING[2]
0037  hex$=''
003E  ascii$=''
0045  FOR n=1 TO c_cnt
0058    RUN make_2hex(m_code(n),h$)
006B    hex$=hex$+h$+' '
007B    IF LAND(m_code(n),$7F)>=$20 THEN
0090      ascii$=ascii$+CHR$(LAND(m_code(n),$7F))
00A5    ELSE
00A9      ascii$=ascii$+'.'
00B5    ENDIF
00B7  NEXT n
00C2  END
```

この部分はBASIC09のLISTコマンドでは欠けてしまいますが，動作には支障ありません

```
PROCEDURE print_source
0000    PARAM loc$:STRING[4]
000C    PARAM hex$:STRING[50]
0018    PARAM ll$:STRING[8]
0024    PARAM m$:STRING[5]
0030    PARAM op$:STRING[50]
003C    PARAM com$:STRING[50]
0048    PARAM ascii$:STRING[20]
0054    PARAM pass_type:INTEGER
005B    PARAM out_fp:INTEGER
0062    PARAM out_mode:INTEGER
0069    PARAM term:BOOLEAN
0070    IF out_mode=1 THEN (* assemble list format *)
0095      PRINT #out_fp,loc$; ' '; TAB(6); hex$; ' '; TAB(24); ll$; ' '
           ; TAB(32); m$; ' '; TAB(38); op$; ' '; TAB(50); com$
           ; ' '; TAB(55); ascii$
00E7    ELSE (* source file format *)
0102      PRINT #out_fp,ll$; ' '; m$; ' '; op$; ' '; com$
0124    ENDIF
0126    IF term THEN (* end of program block *)
0148      PRINT #out_fp,'*'
0152      term=FALSE
0158    ENDIF
015A    END
PROCEDURE find_attr
0000    PARAM a:BYTE
0007    PARAM n:INTEGER
000E    DIM b:BYTE
0015    IF a<>0 THEN
0021      n=1
0028      b=$80
0030      WHILE LAND(a,b)=0 DO
0040        b=INT(b/2)
004D        n=n+1
0058      ENDWHILE
005C      a=LAND(a,LNOT(b))
0069    ELSE (* a=0 *)
0075      n=0
007C    ENDIF
007E    END
PROCEDURE emod
0000    PARAM object_path_no:INTEGER
0007    PARAM address:INTEGER
000E    PARAM term:BOOLEAN
0015    PARAM machine_code_pointer:INTEGER
001C    PARAM machine_code(20):BYTE
0028    PARAM end_flag:BOOLEAN
002F    PARAM operand$:STRING[50]
003B    PARAM comment$:STRING[50]
0047    DIM a:BYTE
004E    RUN get_next_byte(object_path_no,address,a,end_flag)
0067    IF end_flag THEN
0070      END
0072    ENDIF
0074    machine_code(machine_code_pointer+1)=a
0083    machine_code_pointer=machine_code_pointer+1
008E    term=TRUE
0094    operand$=''
009B    END
```

for FM-7,8,11

アマグラマ講座

あなたが作る手作り学習プログラム入門[V]

国重　博

はじめに
学習用ソフトの現状

市販の学習用ソフトも最近ぼつぼつと良いものが発売され，割と使えるものが出回るようになってきています。とくに英語学習用のものは解答入力も容易だし，画面表示もアルファベット中心で見やすく，開発も盛んです。中学用のものは各学校で採択されている教科書別，進度別に編集されたものも出ていますので，実際に成績向上につながるような良質のものも多くなってきていると思います。

ただ，今のところ反復練習型ソフトが中心で思考型・推理型のものが今ひとつなのですが，それはやむを得ないところがあるのではないでしょうか。というのは，今の学校の実情が多かれ少なかれ受験勉強中心ですし，受験勉強というのは結局知識の量が問題になってきます。成績も中学・高校などでは，努力した生徒より知識量の多い生徒のほうに良い点をつけざるを得ないようになっています。ですから，生徒の側にしてもてっとり早く成績を上げるためには知識量をふやすこと，そしてそのためには，暗記と反復練習にウエイトを置くことが，短時間で効果が大きいということになっているようです。

広く浅く知識のある人が成績が良くて，思いがけない人の成績が悪かったりするのは実例に事欠きません。たとえば，外国帰りで英語ペラペラの生徒が，文法の変な規則のところでつまずいて，ペーパテストの成績が今ひとつで，英語の成績が五段階評価の「3」程度だったり，オリンピック級のスポーツ選手が体育の成績が悪かったりなどというのはよくあります。

思考力・創造力も，受験体制の前では二義的なものになってしまっているようです。一流校といわれるような学校（とくに私立中学など）の入試問題は，短時間でたくさんの問題を解かなければならないようになっていますから，思考力テストなどといっても実はその場でじっくり考えているようなヒマはありません。問題を一目見ただけで，解答がピンとくるようでないと高得点はとれないのです。つまり，じっくり考える力ではなく，日頃どれだけ数多くいろいろな問題を解いているか，どれだけトレーニングを積んでいるかによって差がつくようなテストなのです。したがって，一流進学校といわれるような学校へは，家庭教師をつけてもらったり進学塾へ通わせてもらったりしている生徒が多く合格するような傾向になっていると思います。

パソコンを受験勉強中心の学習に役立たせて，直接の効果をあげるために使うのならば，どうしても反復練習型中心にならざるを得ませんが，それはパソコンの責任ではなく，受験体制そのものに問題がありそうです。

受験や成績に直接結びつくというものではないが，思考力を試すというソフトで教育性のあるものとしては，パズル型や思考型のゲームプログラムに結構興味深いものがあります。シミュレーション型のゲームのように，思考力を要求されるゲームでは，成績にかかわらず熱中する生徒と，おもしろさがわかる前に投げ出してしまう生徒がいるようです。ゲームといえどもものによっては単純な教育ソフトよりも教育性がありますが，そういう教育性を学校教育にどう連結させていくかが，これからの課題ではないでしょうか。

いずれにせよ，最近20年間のテレビの普及が日本人の思考パターンを変えたように，パソコンが家庭に真に（ゲームマシンとしてではなく）普及すれば，また社会そのものの思考パターンが変化してくると思います。テレビの普及は娯楽面では一定の役割を果たしているようですが，文化的には，「一億総白痴化」などと言われどうも良い方面には変えなかったようです。しかし，パソコンの普及は，個人のデータ処理能力を飛躍的に向上させてくれるはずです。やがてINS（情報網システム）が進み，いろんなVAN（付加価値通信網）サービスを誰でも受けられるようになれば，どれだけのデータをいかに活用できるかが21世紀の社会的能力の1つとして重視されるようになってくるのではないでしょうか。

アマグラマとしては，細かい小手先のテクニックにとらわれず，近い未来を見通す目を持ち，巨視的な方向性だけは正しく把握しておくように心がけようではありませんか。

1.グラフィック命令

前置きが長くなりましたが，実際のプログラム例をみていくことにしましょう。今回はグラフィック関係の命令を使った学習プログラムに取り組んでみます。

グラフィック関係の命令は，主なものとして，

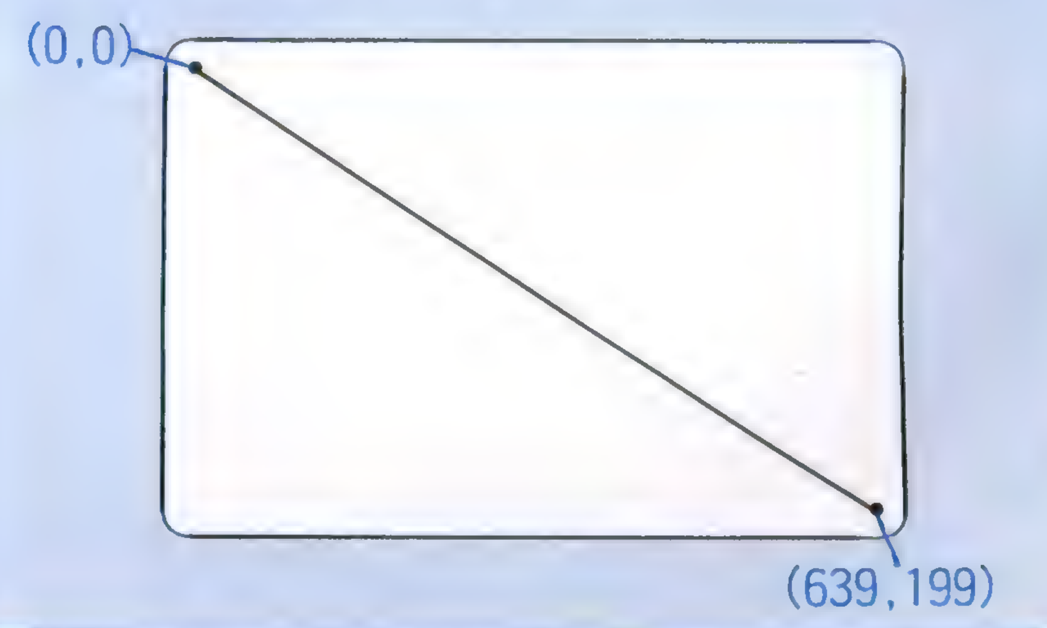
LINE（ライン） 直線または長方形を描く
CIRCLE（サークル） 円・楕円を描く
CONNECT（コネクト） 点から点へ線を結ぶ

リスト1-1

```
10 CLS
20 LINE (0,0)-(639,199),PSET,6
30 END
```

カラーコード6は黄色です

図1-1　リスト1-1の実行例



PAINT（ペイント）　色を塗る

などがありますが，これで折れ線グラフや棒グラフなどのグラフを描いたり，図を描いたりすることができます。さらに，GET@やPUT@，PSET，パレット機能などを活用すると，ディスプレイをキャンバス代わりにして，C.G.を楽しんだり，アニメを作ったりできるわけです。

各命令の詳しい内容はマニュアルを参照していただくとして，グラフィックスを描くときに留意しなければならない点をいくつかみていきましょう。

a.**リスト1―1**とその実行例（図1―1）をみてください。画面の左上スミ〔座標（0，0）〕から右下スミ〔座標（639,199）〕へ黄色〔カラーコード6〕で線が引かれます。20行目の(0,0)―(639,199)に注意してください。グラフィック命令では，ドット単位の座標を指定する必要があります。

b.**リスト1―2**とその実行例(図1―2)では，（0，0）と（639，199)を対角線の頂点とする長方形を描いています。LINE文は，直線を描くだけでなく，長方形を描いたり塗りつぶしたりすることもできるのです。

c.**リスト1―3**は，(0,0)―(100,100)の四角形を描きます。横101ドット，縦101ドットと一辺のドット数が同じだから正方形になりそうなものですが，ディスプレイのドット数の縦横の比率が違うのでずいぶん縦長の長方形になります。

d.**リスト1―4**では，比率を調整しています。横，縦のドット比率は約20:9なので，それに合わせて座標をとると，ほぼ正方形になります。つまり，座標をとっていきながら図を描くときには，いびつにならないように横縦のドット比率に常に留意する必要があります。

e.**リスト1―5**は，**リスト1―4**の正方形をPAINT文で塗りつぶしています。色を赤(2)，境界色を緑(4)に指定したので，ワクが緑で中が赤になっています。**リスト1―4**の最後をBFにするとやはり塗りつぶした四角形が描けますが，そのときはワクと中味が同じ色になってしまいます。

以上のことをまとめると，次のようになります。

（1）グラフィック命令は，要するにディスプレイの画面をドット単位で表現するための命令です。

（2）ドット数は，横640，縦200の計128000個あります。この一つひとつにカラーを指定して，C.G.を描いたりすることが可能です。

（3）ディスプレイ上では，ドット比率が横：縦が約20:9なので，図形を描くときにはその分を調整してやる必要があります。

（4）機能の部分は，図の重なりや線の交点の色などに影響します。

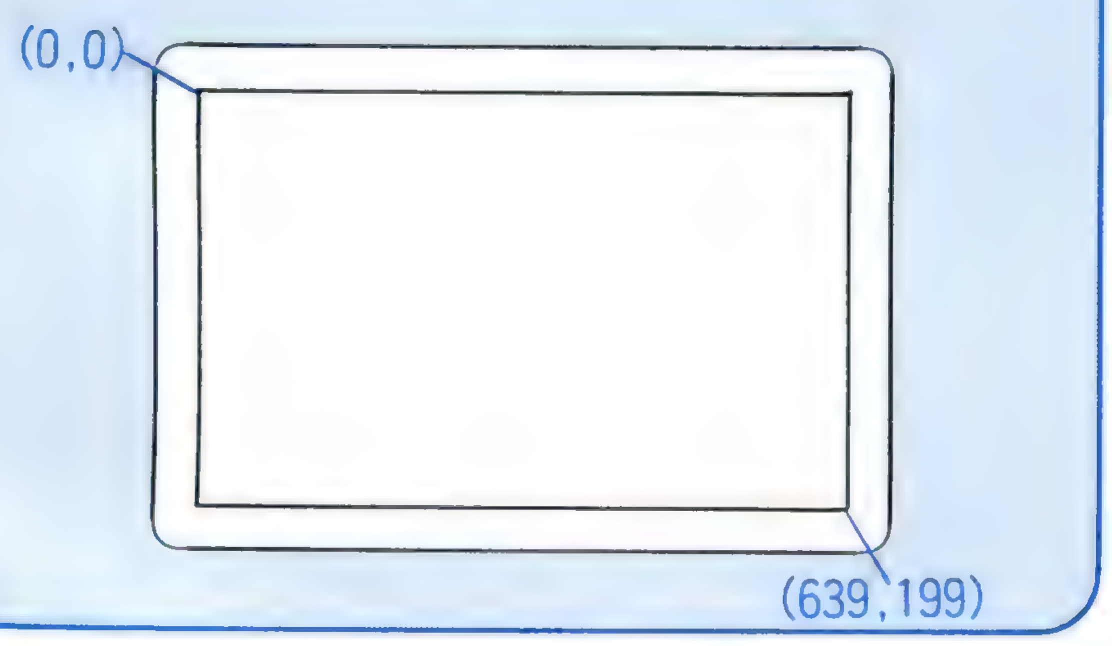
これらのことに留意して，実際にプログラミングしていくことにしましょう。

リスト1-2

```
10 CLS
20 LINE (0,0)-(639,199),PSET,6,B
30 END
```

長方形を描きます

図1-2　リスト1-2の実行例

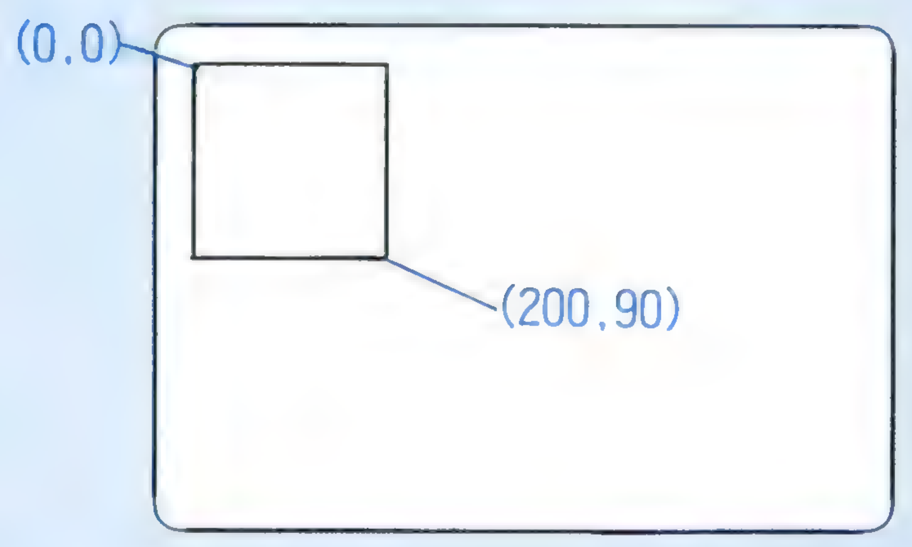


リスト1-4

```
10 CLS
20 LINE (0,0)-(200,90),PSET,4,B
30 END
```

横と縦を20:9にすると，ほぼ正方形になります

図1-4　リスト1-4の実行例

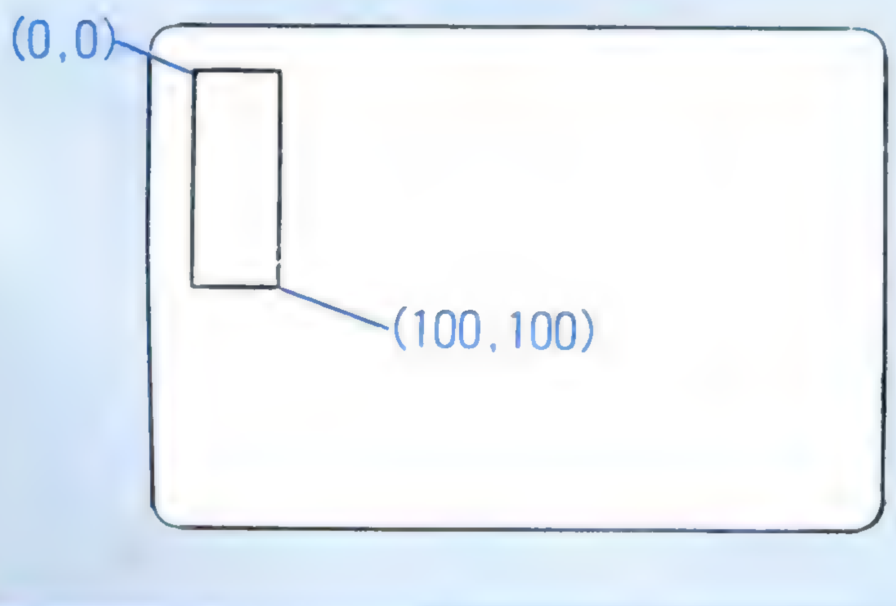


リスト1-3

```
10 CLS
20 LINE (0,0)-(100,100),PSET,5,B
30 END
```

横，たて共座標は100なのに正方形になりません

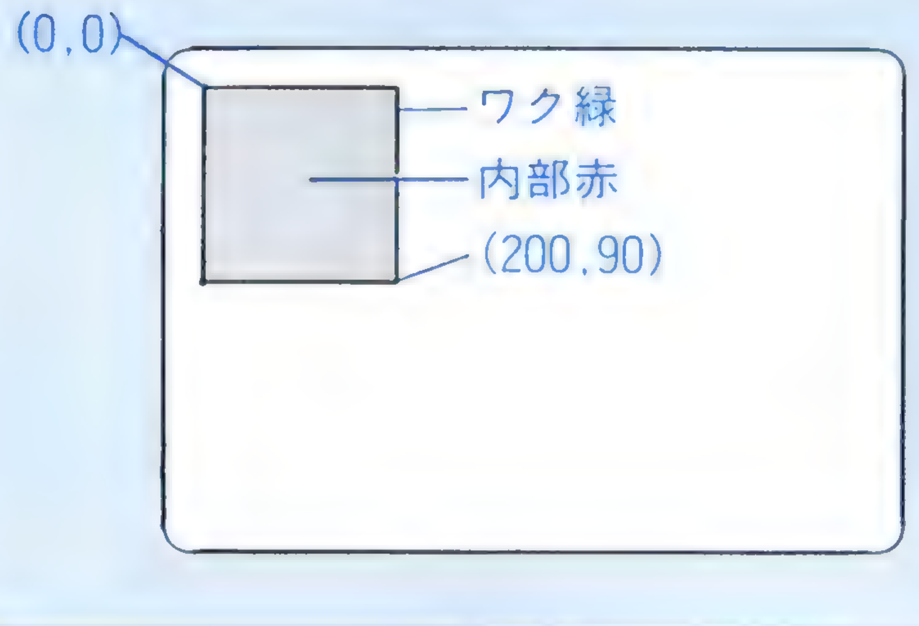
図1-3　リスト1-3の実行例



リスト1-5

```
10 CLS
20 LINE (0,0)-(200,90),PSET,4,B
30 PAINT (50,50),2,4
40 END
```

図1-5　リスト1-5の実行例



2.線描サブルーチン

リスト2は，CONNECT文とREAD～DATAを使って，線描図を描くプログラムです。グラフィック命令をいろいろ使って実際にディスプレイ上で絵を描いてみると，CONNECT文が意外に便利で実用的だと思われます。**リスト2**は北海道の地図を描くプログラムで，独立した形になっていますが，ふつうこれをサブルーチンの形にしておくと，いろんな用途に活用できると思います。主な座標だけをデータにしておいて線描し，必要に応じてPAINTすれば結構いろいろな絵が描けますし，画面いっぱいの大きな絵のときでも，GET@，PUT@などに比してメモリが少なくてすみます。RESTORE文でDATAの先頭を指定してサブルーチンに飛ぶことによって，任意の図が描けますし，場合によっては，LINE文やCI

RCLE文を使わなくても代用できるほどです。CIRCLE文は，円グラフなどを描くときは便利ですが，半径の小さな円を描くときは，ドットの関係で円周のなめらかさがなくなってきます。半径11程度だと，円というより六角形です。それにCIRCLEは中心の座標は把握できても，円周部の座標を把握するのはわりと面倒で，部分的に使ってつないでいくのは手間がかかるのです。

リスト2のプログラムの構造は，30行目のREAD文で，10個ずつ（つまり5個の座標）のデータを読み取り，順次40行目のCONNECT文でつないでいくようになっています。北海道の場合，データ数が70個ですので，それを7回繰り返しています。

データをいろいろ変えると，地図だけではなくいろいろな図を表示することができます。このプログラムのデータは，画面の比率に合わせて座標をとっていますが，方眼紙に絵を書いて，データ化する場合は，読み取ったデータの縦座標の値に$\frac{9}{20}$，または0.4495をかけてから，CONNECTするとゆがみのない図になります。

カラーをいろいろ変えてみたり，データをいろいろ処理してからCONNECTするようにすると，拡大・縮小，位置自由，反転なども可能です。三角関数と組み合わせると三次元グラフィックスにもなるわけです。最初のデータ取りが少しやっかいですが，それをこなしてしまうとかなり自由に，機能的に使えると思います。

3.日本地図表示プログラム

リスト3は，**リスト2**のプログラムをサブルーチン化して，日本地図をいろんなふうに描けるようにしたプログラムです。

RUNさせると，30〜90行でメニュが表示されます。どれかを指定すると，180行のON〜GOTO文で190〜240行に飛ぶわけですが，その前に130〜160行で位置や拡大率，カラーなどを選べるようになっています。

メニュ1で，全体を選び，位置(0，0)拡大率1，カラーは任意で実行し，日本地図を描いてみてください。この図が基本図になっています。この図をもとにして，(0，0)を起点として拡大・縮小，位置決めをしています。これだけで，縦にはぼいっぱいにとっているので，実は全体図などはほとんど拡大はできません。縮小(拡大率 0〜1)は自由です。位置なども，(0，0)〜(639，199)のワクをはみ出してしまうと，形が崩れたりエラーが出たりします。拡大率や位置の許容範囲は，メニュによって異なるのですが，試行錯誤的にためしてみてください(実行例参照)。

データ部が多くて恐縮ですが，**リスト2，3，4**のデータは共通して使えますので，順次追加していけばよいようになっています。**リスト3**のデータを入力すれば，**リスト4**の入力は楽になりますので頑張ってください。

リスト2

©1984 H.KUNISHIGE

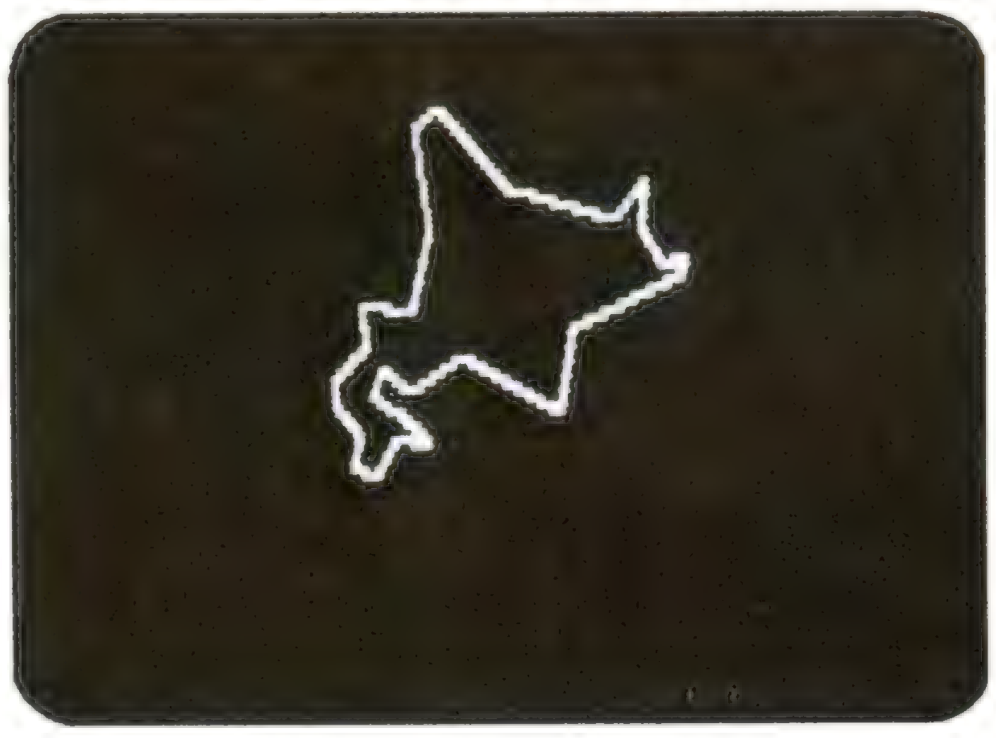
```
10 ' ﾎｯｶｲﾄﾞｳ
20 FOR I=1TO 7
30 READ A1,A2,A3,A4,A5,A6,A7,A8,A9,A10
40 CONNECT (A1,A2)-(A3,A4)-(A5,A6)-(A7,A8)-(A9,A10),1
50 NEXT I
60 END
70 DATA 300,0,292,2,295,17,288,28,270,27
80 DATA 270,27,271,33,259,37,259,41,267,45
90 DATA 267,45,264,50,272,51,280,46,291,47
100 DATA 291,46,280,42,272,40,276,36,285,39
110 DATA 286,39,305,35,339,43,345,31,384,25
120 DATA 384,25,386,22,378,23,371,18,373,11
130 DATA 373,11,362,16,326,12,300,0,300,0
```

20行：データ数が70ですので，7度繰り返します
30行：データは10個ずつ読み，つなぐ座標は5個ずつです
40行：1は青色です
70〜130行：座標は35ですのでデータ数は70です
130行の「300,0」：座標は34で済んだのですがデータ数を合わせるために同じものを2つ続けています

○BASIC
CONNECT（コネクト）座標を順次つないでいきます。

○改造のポイント
(1)データを変えるといろいろな図が描けます。
PAINT文と組み合わせると5Kバイト分ほどのメモリでかなり複雑な絵を描くことができます。
(2)サブルーチンとして活用できます。

写真1　リスト2の実行例（写真拡大）



BASICチェックサム

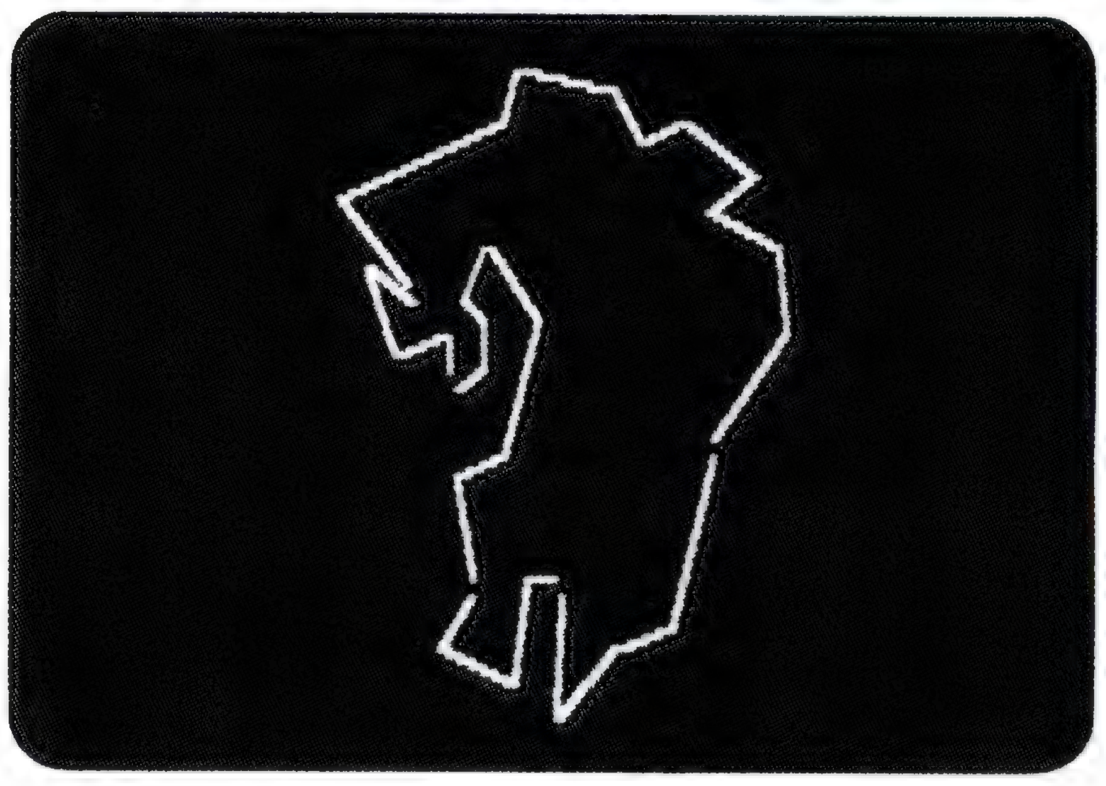
```
LINE +0 +1 +2 +3 +4 +5 +6 +7 SUM
  10 00 78 70 54 88 D7 4E BF :A8
  90 BB BE BE BB 2B 00 00 00 :1D
   0 00 00 00 00 00 00 00 00 :00
   0 00 00 00 00 00 00 00 00 :00
   0 00 00 00 00 00 00 00 00 :00
   0 00 00 00 00 00 00 00 00 :00
   0 00 00 00 00 00 00 00 00 :00
   0 00 00 00 00 00 00 00 00 :00
   0 00 00 00 00 00 00 00 00 :00
   0 00 00 00 00 00 00 00 00 :00
--------------------------------
     BB 36 2E 0F B3 D7 4E BF /C5
```

リスト3　日本地図表示プログラム

```
10 ´ﾁｽﾞ
20 WIDTH40,20
30 PRINT "ﾆﾎﾝﾁｽﾞ ｦ ｶｷﾏｽ｡ 1-6 ｦ ｴﾗﾝﾃﾞｸﾀﾞｻｲ"
40 PRINT "ｾﾞﾝﾀｲ------1"
50 PRINT "ﾎｯｶｲﾄﾞｳ----2"
60 PRINT "ﾎﾝｼｭｳ------3"
70 PRINT "ｼｺｸ--------4"
80 PRINT "ｷｭｳｼｭｳ-----5"
90 PRINT "ﾄﾄﾞｳﾌｹﾝ----6"
100 PRINT
110 INPUT "1-6 ｦ ｴﾗﾌﾞ";P
120 PRINT
130 PRINT "ｲﾁ"
140 INPUT "ﾖｺ---";V:INPUT"ﾀﾃ---";W
150 INPUT "ｶｸﾀﾞｲﾘﾂ";K
160 INPUT "ｶﾗｰ･･･1-7";CO
170 CLS
180 ON P GO TO 190,200,210,220,230,240
190 Q=50:RESTORE 390:GOSUB260:GO TO 250:´ｾﾞﾝﾀｲ 50
200 Q=7:RESTORE 390 :GOSUB260:GO TO 250:´ﾎｯｶｲﾄﾞｳ 7
210 Q=21:RESTORE 470:GOSUB260:GO TO 250:´ﾎﾝｼｭｳ 21
220 Q=5:RESTORE 690 :GOSUB260:GO TO 250:´ｼｺｸ 5
230 Q=8:RESTORE 750:GOSUB260:GO TO 250:´ｷｭｳｼｭｳ 8
240 Q=88:RESTORE 390:GOSUB260:GO TO 250:´ｹﾝｻﾞｶｲ 38+50
250 END
260 ´ ｸﾞﾗﾌｨｯｸ ｻﾌﾞ
270 FOR I=1TO Q
280 READ A1,A2,A3,A4,A5,A6,A7,A8,A9,A10
290 A1=(A1+V)*K:A2=(A2+W)*K
300 A3=(A3+V)*K:A4=(A4+W)*K
310 A5=(A5+V)*K:A6=(A6+W)*K
320 A7=(A7+V)*K:A8=(A8+W)*K
330 A9=(A9+V)*K:A10=(A10+W)*K
340 CONNECT (A1,A2)-(A3,A4)-(A5,A6)-(A7,A8)-(A9,A10),CO
350 NEXT I
360 RETURN
370 ´ﾃﾞｰﾀ
380 ´ ﾎｯｶｲﾄﾞｳ
390 DATA 300,0,292,2,295,17,288,28,270,27
400 DATA 270,27,271,33,259,37,259,41,267,45
410 DATA 267,45,264,50,272,51,280,46,291,47
420 DATA 291,46,280,42,272,40,276,36,285,39
430 DATA 286,39,305,35,339,43,345,31,384,25
440 DATA 384,25,386,22,378,23,371,18,373,11
450 DATA 373,11,362,16,326,12,300,0,300,0
460 ´ﾎﾝｼｭｳ
470 DATA 287,49,284,53,292,52,291,57,283,57
480 DATA 283,57,280,54,273,54,272,58,265,60
490 DATA 265,60,266,68,262,69,263,71,269,70
500 DATA 269,70,270,80,250,95,218,109,204,112
510 DATA 204,112,200,107,207,102,192,104,191,114
520 DATA 191,114,174,122,175,126,167,129,157,128
530 DATA 157,128,159,125,112,129,98,128,51,142
540 DATA 51,142,50,148,82,149,87,143,91,145
550 DATA 91,145,125,141,142,138,163,139,155,144
560 DATA 155,144,157,150,165,154,176,154,192,145
570 DATA 192,145,200,144,199,140,191,138,196,134
580 DATA 196,134,198,138,200,136,208,137,202,140
590 DATA 202,140,232,138,243,132,248,134,245,137
600 DATA 245,137,249,139,255,136,254,132,260,130
610 DATA 261,130,267,131,272,125,280,126,273,130
620 DATA 273,131,274,134,287,130,295,123,290,119
630 DATA 290,119,297,106,295,96,292,92,300,88
640 DATA 301,88,306,90,315,78,312,65,300,58
650 DATA 299,58,298,50,287,49,287,49,287,49
660 DATA 234,90,226,95,230,96,227,99,237,95
670 DATA 237,95,233,94,237,91,234,90,234,90
680 ´ｼｺｸ
690 DATA 126,143,117,145,118,147,104,148,100,146
700 DATA 100,146,87,153,77,156,85,155,90,163
710 DATA 90,163,101,164,112,156,121,155,130,157
720 DATA 130,157,144,149,141,144,126,143,126,143
730 DATA 152,140,144,143,144,145,151,144,153,141
740 ´ｷｭｳｼｭｳ
750 DATA 50,149,36,148,34,151,10,155,20,160
760 DATA 21,161,14,159,18,164,26,163,26,165
770 DATA 27,166,31,165,31,162,28,161,31,158
780 DATA 32,158,39,162,34,170,27,171,29,177
790 DATA 29,178,24,181,35,183,37,177,41,177
800 DATA 42,178,41,185,50,181,58,180,64,170
810 DATA 64,169,74,163,73,158,63,156,70,154
```

(注釈)
- 40〜90行: メニュ表示
- 100行: 1行あける
- 110行: メニュ選択
- 120行: 1行あける
- 140行: 位置指定
- 150行: 拡大率
- 160行: カラー
- 190〜240行: メニュにより，データの位置と数を指定して，サブルーチンに飛びます。最後の数字はデータ数です(×10)
- 290〜330行: 5個の座標の位置をずらして拡大率をかけます
- 390〜450行: リスト2の70〜130行

写真2 リスト3の実行例



```
820 DATA 70,154,60,151,54,152,50,149,50,149
830 'ｼﾏ
840 DATA 144,199,144,185,252,155,276,155,276,199
850 DATA 224,174,219,175,214,175,213,178,209,180
860 DATA 210,181,208,184,214,184,215,181,217,181
870 DATA 217,180,216,179,228,175,224,174,224,174
880 DATA 11,137,7,138,5,142,11,142,12,138
890 DATA 9,144,4,144,5,146,11,146,10,144
900 DATA 48,188,42,193,44,195,47,193,49,189
910 DATA 36,192,30,192,29,194,33,196,37,193
920 DATA 183,128,179,128,178,134,185,132,183,128
930 ' ｹﾝ ｻﾞｶｲ
940 DATA 266,63,288,65,287,82,288,65,305,62
950 DATA 270,80,284,83,287,82,300,83,313,80
960 DATA 284,83,280,95,278,94,291,97,294,91
970 DATA 278,94,276,97,263,96,266,91,259,88
980 DATA 267,97,253,103,256,109,272,106,281,110
990 DATA 294,109,281,110,280,114,268,118,270,120
1000 DATA 295,123,288,121,280,122,270,120,273,125
1010 DATA 260,108,257,112,262,113,259,116,268,118
1020 DATA 253,107,247,110,240,111,238,108,220,111
1030 DATA 268,118,252,118,244,122,250,123,250,123
1040 DATA 273,123,259,123,258,122,250,123,256,125
1050 DATA 273,127,256,125,251,128,253,132,253,132
1060 DATA 240,111,234,115,240,117,244,122,238,123
1070 DATA 238,123,231,122,232,129,240,130,251,128
1080 DATA 218,108,219,111,212,124,218,131,195,130
1090 DATA 232,128,226,131,218,130,222,131,214,139
1100 DATA 201,110,197,112,198,117,209,116,217,117
1110 DATA 182,118,195,120,197,118,194,125,185,126
1120 DATA 196,131,193,132,194,134,193,132,189,132
1130 DATA 171,130,185,126,189,132,187,136,178,137
1140 DATA 165,128,171,130,177,137,176,138,170,137
1150 DATA 177,139,180,146,176,149,174,149,180,152
1160 DATA 148,126,149,127,152,127,153,129,150,130
1170 DATA 151,131,160,133,161,138,160,133,169,136
1180 DATA 169,136,167,143,156,144,156,144,156,144
1190 DATA 167,143,168,144,167,146,169,149,174,148
1200 DATA 136,127,139,131,134,136,136,138,136,138
1210 DATA 136,132,120,131,111,134,106,133,111,129
1220 DATA 115,142,111,139,106,134,84,138,81,142
1230 DATA 68,138,74,142,80,142,85,145,85,145
1240 DATA 116,147,122,148,121,150,122,148,140,145
1250 DATA 134,154,129,151,115,150,96,157,93,163
1260 DATA 54,152,54,154,45,155,46,158,32,159
1270 DATA 27,153,34,156,30,158,30,158,30,158
1280 DATA 17,154,22,158,29,160,29,160,29,160
1290 DATA 46,158,48,160,51,159,57,163,72,164
1300 DATA 57,164,50,167,49,172,44,173,29,171
1310 DATA 44,173,47,177,52,178,53,181,53,181
```

○改造のポイント

メインプログラムはかなり簡単ですので手を加える余地があります。

(1)位置，拡大率の数字によっては，すぐにワクをはみ出したり、エラーが出たりしますので，調整できるようにしてみてください。

(2)横座標に－１をかけて，位置をずらすと左右反転します。縦座標だと，上下反転です。
両方とも負にすると，上下左右反転像が出ます。

(3)プリンタのある人は，ハードコピーをとれるようにしてみてください。

BASICチェックサム

```
LINE +0 +1 +2 +3 +4 +5 +6 +7 SUM
  10 00 72 40 B7 AD 95 57 D6 :D8
  90 F2 8D CA 8D 00 9E 6E 1E :00
 170 E2 80 83 55 80 56 56 8E :F4
 250 D7 00 92 70 0D 15 1D 25 :3D
 330 7B B5 88 E0 00 00 4E BF :A5
 410 BB BE BE BB 2B 00 CF C2 :AE
 490 C9 13 88 B0 4E B8 76 B1 :41
 570 AE A6 9D AB 9D AD 2F B5 :CA
 650 E9 CB BF 00 9E EF 66 9F :05
 730 96 00 A2 AF AD B9 C2 B6 :C5
---------------------------------
     D7 76 EB AE 9B AB 22 E3 /31

LINE +0 +1 +2 +3 +4 +5 +6 +7 SUM
 810 C3 AB 00 C9 B1 A4 B4 3A :7A
 890 0C D4 C0 BC 00 D2 B8 DA :C0
 970 E5 79 A2 A1 AA 8D 9C A4 :18
1050 A4 90 94 98 95 A1 B5 AF :FA
1130 B7 B1 BD A5 9C B5 C4 AA :89
1210 86 3B BF 97 4A BA AE B0 :79
1290 C0 BE BE 00 00 00 00 00 :3C
---------------------------------
     55 32 30 FA D6 13 2F C1 /8A
```

4.日本の都道府県

リスト4は，日本の都道府県名を学習するプログラムです。サブルーチンの部分などは**リスト3**と共通ですが，今度は乱数を使って出題するので，30行で配列変数を用意しています。D１（Ｉ）は横座標，D２（Ｉ）は縦座標，そしてＥ＄（Ｉ）には都道府県名が入ります。90行で乱数が設定され，たとえばＲ＝１のとき，Ｅ＄（R）はＥ＄（1）ですから〝ホッカイドウ〟です。D1(1)，D2(1)は30，25です。Ｅ＄（２）は青森県，Ｅ＄（３）は秋田県と続きます。100～120行では，質問が表示されます。画面中央部上に，〝ココハ？〟と出て，問題の県が赤く塗られ，線で指示されます。130行のINPUTで解答入力を要求して，正解のときには効果音サブルーチンへ飛んで，得点が10点プラスされます。10問で一区切りで，全問正解だと得点は100点です。200～220行は，得点によってメッセージが出るようにしています。

おもしろいのは，120，160行のLINE文におけるXOR（排他的論理和）の使い方で機能をXORにしておくと，２度線を引くとその線が消えてしまいます。他の線と交わっているとき，同じ色だと交点は消え，他の色だと色が変わりますが，２度目でもとに戻ります。PSETで線を引き，PRESETでその線を消すと，他の線との交点も消してしまうので，PAINTした時などそこから色がもれてしまったりするのですが，XORだとそんなことにはなりません。

このプログラムはデータ部が長いのですが，メイン部は20行ほどです。いろいろ手を加える余地があると思います。また，データを変えることによって，世界地図にしたり，山地，山脈，平野，河川，県庁所在地など，各種の応用がきくはずです。

編物の上手な女性などは，たんねんにデータをとって，すばらしい絵が描けるのではないでしょうか。プログラミングも性格を反映するようで，男性向きのルーチンと女性向きのルーチンがあるようですね。まあ編物が趣味という男性もいますから一概には言えませんが…。

次回は，漢字表示に挑戦する予定です。

リスト4　日本の都道府県(小学5年～中学2年)

```
10 'ﾆﾎﾝﾁﾘ
20 CLEAR:RANDOMIZE TIME
30 DIM D1(47),D2(47),E$(47)
40 WIDTH40,20
50 RESTORE 390:Q=50:CO=1:GOSUB 300:'ﾜｸ
60 RESTORE 940:Q=38:GOSUB 300:'ｻｶｲ
70 RESTORE1320 :FOR I=1 TO 47:READ D1(I),D2(I):NEXT I:'ｲﾁ
80 RESTORE1350 :FOR I=1 TO 47:READ E$(I):NEXT I:'ﾅﾏｴ
90 R=INT(RND*47)+1:NO=NO+1
100 BEEP1:BEEP0:LOCATE20,3:PRINT NO;"ｺｺﾊ?"
110 PAINT (D1(R)+V,D2(R)),2,1
120 LINE (400,50)-(D1(R)+V,D2(R)),XOR,4
130 LOCATE 0,2:INPUT ANS$
140 IF ANS$=E$(R) THEN GOSUB 250 ELSE BEEP:SC=SC-5:GO TO 130
150 SC=SC+10
160 LINE (400,50)-(D1(R)+V,D2(R)),XOR,4
170 PAINT (D1(R)+V,D2(R)),0,1
180 LOCATE0,0 :PRINT "SCORE:";SC
190 IF NO<10 THEN 90
200 GOSUB240:IF SC>=90 THEN PRINT"ﾖｸﾃﾞｷﾏｼﾀ!!":GO TO 230
210 IF SC>=60 THEN PRINT"ﾓｳｽｺｼ ｶﾞﾝﾊﾞﾚ!!":GO TO 230
220 PRINT"ｼｯｶﾘ ｼﾅｻｲ!!"
230 END
240 'ｺｳｶｵﾝ
250 FOR I=10 TO 5 STEP -1
260  FOR J=0 TO I:BEEP 1:NEXT J
270  FOR J=I TO 3 STEP -1:BEEP 0:NEXT J
280 NEXT I
290 RETURN
300 ' ｸﾞﾗﾌｨｯｸ ｻﾌﾞ
310 V=250
320 FOR I=1TO Q
330 READ A1,A2,A3,A4,A5,A6,A7,A8,A9,A10
340 CONNECT (A1+V,A2)-(A3+V,A4)-(A5+V,A6)-(A7+V,A8)-(A9+V,A10),CO
350 NEXT I
360 RETURN
370 'ﾃﾞｰﾀ
380 ' ﾎｯｶｲﾄﾞｳ
390 DATA 300,0,292,2,295,17,288,28,270,27
400 DATA 270,27,271,33,259,37,259,41,267,45
410 DATA 267,45,264,50,272,51,280,46,291,47
420 DATA 291,46,280,42,272,40,276,36,285,39
430 DATA 286,39,305,35,339,43,345,31,384,25
440 DATA 384,25,386,22,378,23,371,18,373,11
450 DATA 373,11,362,16,326,12,300,0,300,0
460 'ﾎﾝｼｭｳ
470 DATA 287,49,284,53,292,52,291,57,283,57
480 DATA 283,57,280,54,273,54,272,58,265,60
490 DATA 265,60,266,68,262,69,263,71,269,70
500 DATA 269,70,270,80,250,95,218,109,204,112
510 DATA 204,112,200,107,207,102,192,104,191,114
520 DATA 191,114,174,122,175,126,167,129,157,128
530 DATA 157,128,159,125,112,129,98,128,51,142
540 DATA 51,142,50,148,82,149,87,143,91,145
550 DATA 91,145,125,141,142,138,163,139,155,144
560 DATA 155,144,157,150,165,154,176,154,192,145
570 DATA 192,145,200,144,199,140,191,138,196,134
580 DATA 196,134,198,138,200,136,208,137,202,140
590 DATA 202,140,232,138,243,132,248,134,245,137
600 DATA 245,137,249,139,255,136,254,132,260,130
610 DATA 261,130,267,131,272,125,280,126,273,130
620 DATA 273,131,274,134,287,130,295,123,290,119
630 DATA 290,119,297,106,295,96,292,92,300,88
640 DATA 301,88,306,90,315,78,312,65,300,58
650 DATA 299,58,298,50,287,49,287,49,287,49
660 DATA 234,90,226,95,230,96,227,99,237,95
670 DATA 237,95,233,94,237,91,234,90,234,90
680 'ｼｺｸ
690 DATA 126,143,117,145,118,147,104,148,100,146
700 DATA 100,146,87,153,77,156,85,155,90,163
710 DATA 90,163,101,164,112,156,121,155,130,157
```

- 30: 都道府県の位置と県名の配列変数です
- 50～80: 地図を描きます
- 90: Rは1～47までの乱数，NOは問題番号です
- 110: 問題の都道府県を赤く塗ります
- 130: 解答入力
- 140: 誤答のときは−5
- 150: 正解だと＋10です
- 160: XORで2度線を引くと消えます
- 170: 都道府県の色を消します
- 180: 得点の表示
- 190: 10問になるまで，90に戻ります
- 200～220: 得点によって，メッセージが出ます
- 240: ピコピコというサブルーチンです
- 250～280: 7,77の人はメロディを考えてください
- 310: 基本位置から250ドット横にずらします
- 320～350: 指定の回数(Q)だけデータを読んで，線を引きます
- 370～710: リスト3の370～710と同じ

写真3　リスト4の実行例

リスト3の720～1310と同じ

```
720 DATA 130,157,144,149,141,144,126,143,126,143
730 DATA 152,140,144,143,144,145,151,144,153,141
740 'ｷｭｳｼｭｳ
750 DATA 50,149,36,148,34,151,10,155,20,160
760 DATA 21,161,14,159,18,164,26,163,26,165
770 DATA 27,166,31,165,31,162,28,161,31,158
780 DATA 32,158,39,162,34,170,27,171,29,177
790 DATA 29,178,24,181,35,183,37,177,41,177
800 DATA 42,178,41,185,50,181,58,180,64,170
810 DATA 64,169,74,163,73,158,63,156,70,154
820 'ｼﾏ
830 DATA 70,154,60,151,54,152,50,149,50,149
840 DATA 144,199,144,185,252,155,276,155,276,199
850 DATA 224,174,219,175,214,175,213,178,209,180
860 DATA 210,181,208,184,214,184,215,181,217,181
870 DATA 217,180,216,179,228,175,224,174,224,174
880 DATA 11,137,7,138,5,142,11,142,12,138
890 DATA 9,144,4,144,5,146,11,146,10,144
900 DATA 48,188,42,193,44,195,47,193,49,189
910 DATA 36,192,30,192,29,194,33,196,37,193
920 DATA 183,128,179,128,178,134,185,132,183,128
930 ' ｹﾝ ｻﾞｶｲ
940 DATA 266,63,288,65,287,82,288,65,305,62
950 DATA 270,80,284,83,287,82,300,83,313,80
960 DATA 284,83,280,95,278,94,291,97,294,91
970 DATA 278,94,276,97,263,96,266,91,259,88
980 DATA 267,97,253,103,256,109,272,106,281,110
990 DATA 294,109,281,110,280,114,268,118,270,120
1000 DATA 295,123,288,121,280,122,270,120,273,125
1010 DATA 260,108,257,112,262,113,259,116,268,118
1020 DATA 253,107,247,110,240,111,238,108,220,111
1030 DATA 268,118,252,118,244,122,250,123,250,123
1040 DATA 273,123,259,123,258,122,250,123,256,125
1050 DATA 273,127,256,125,251,128,253,132,253,132
1060 DATA 240,111,234,115,240,117,244,122,238,123
1070 DATA 238,123,231,122,232,129,240,130,251,128
1080 DATA 218,108,219,111,212,124,218,131,195,130
1090 DATA 232,128,226,131,218,130,222,131,214,139
1100 DATA 201,110,197,112,198,117,209,116,217,117
1110 DATA 182,118,195,120,197,118,194,125,185,126
1120 DATA 196,131,193,132,194,134,193,132,189,132
1130 DATA 171,130,185,126,189,132,187,136,178,137
1140 DATA 165,128,171,130,177,137,176,138,170,137
1150 DATA 177,139,180,146,176,149,174,149,180,152
1160 DATA 148,126,149,127,152,127,153,129,150,130
1170 DATA 151,131,160,133,161,138,160,133,169,136
1180 DATA 169,136,167,143,156,144,156,144,156,144
1190 DATA 167,143,168,144,167,146,169,149,174,148
1200 DATA 136,127,139,131,134,136,136,138,136,138
1210 DATA 136,132,120,131,111,134,106,133,111,129
1220 DATA 115,142,111,139,106,134,84,138,81,142
1230 DATA 68,138,74,142,80,142,85,145,85,145
1240 DATA 116,147,122,148,121,150,122,148,140,145
1250 DATA 134,154,129,151,115,150,96,157,93,163
1260 DATA 54,152,54,154,45,155,46,158,32,159
1270 DATA 27,153,34,156,30,158,30,158,30,158
1280 DATA 17,154,22,158,29,160,29,160,29,160
1290 DATA 46,158,48,160,51,159,57,163,72,164
1300 DATA 57,164,50,167,49,172,44,173,29,171
1310 DATA 44,173,47,177,52,178,53,181,53,181
1320 ' ｲﾁ
1330 DATA 320,25,283,58,280,73,294,72,270,88,290,91,279,102,252,115,270,113,285,
115,261,121,270,125,260,128,286,125,250,100,228,120,240,126,234,135,204,133,204,
113,204,125,192,115,180,123
1340 DATA 182,135,172,142,184,142,162,149,164,139,164,132,150,135,125,129,124,13
9,95,130,96,140,62,145,127,144,138,149,114,153,96,150,36,152,64,158,60,174,29,15
7,19,162,42,165,34,178,216,177
1350 DATA ﾎｯｶｲﾄﾞｳ,ｱｵﾓﾘｹﾝ,ｱｷﾀｹﾝ,ｲﾜﾃｹﾝ,ﾔﾏｶﾞﾀｹﾝ,ﾐﾔｷﾞｹﾝ,ﾌｸｼﾏｹﾝ,ｸﾞﾝﾏｹﾝ,ﾄﾁｷﾞｹﾝ,ｲﾊﾞﾗｷｹﾝ
,ｻｲﾀﾏｹﾝ,ﾄｳｷｮｳﾄ,ｶﾅｶﾞﾜｹﾝ,ﾁﾊﾞｹﾝ,ﾆｲｶﾞﾀｹﾝ,ﾅｶﾞﾉｹﾝ,ﾔﾏﾅｼｹﾝ,ｼｽﾞｵｶｹﾝ,ｱｲﾁｹﾝ,ﾄﾔﾏｹﾝ,ｷﾞﾌｹﾝ,ｲｼｶ
ﾜｹﾝ,ﾌｸｲｹﾝ,ｼｶﾞｹﾝ,ﾅﾗｹﾝ,ﾐｴｹﾝ,ﾜｶﾔﾏｹﾝ
1360 DATA ｵｵｻｶﾌ,ｷｮｳﾄﾌ,ﾋｮｳｺﾞｹﾝ,ﾄｯﾄﾘｹﾝ,ｵｶﾔﾏｹﾝ,ｼﾏﾈｹﾝ,ﾋﾛｼﾏｹﾝ,ﾔﾏｸﾞﾁｹﾝ,ｶｶﾞﾜｹﾝ,ﾄｸｼﾏｹﾝ,ｺ
ｳﾁｹﾝ,ｴﾋﾒｹﾝ,ﾌｸｵｶｹﾝ,ｵｵｲﾀｹﾝ,ﾐﾔｻﾞｷｹﾝ,ｻｶﾞｹﾝ,ﾅｶﾞｻｷｹﾝ,ｸﾏﾓﾄｹﾝ,ｶｺﾞｼﾏｹﾝ,ｵｷﾅﾜｹﾝ
```

○BASIC

PAINT(ペイント)…境界線で囲まれた範囲を指定色で塗ります。

LINE(ライン), XOR(排他的論理和)

LINE文で線を引くとき，線の重なり部の色と機能を活用しましょう。

(1)機能をXORにして，2度線を引くと2度目で消えます。

(2)青の上にXORで緑の線を交わらせると，交点は水色になります。2度目で交点はもとの青に戻ります。

○学習のポイント

都道府県は小学5年生で学習しますが，中学生でも結構できないようです。

○改造のポイント

(1)カタカナ入力が少しわずらわしいので，選択式で解答できるようにしてみてください。

(2)都道府県の位置の座標は，ほぼ県庁所在地にとってあります。県庁所在都市も答えるように改造してみてください。

(3)データ部さえ変えれば，山地・山脈・平野・河川なども出題できるようになります。

BASICチェックサム

LINE	+0	+1	+2	+3	+4	+5	+6	+7	SUM
10	00	79	BA	72	AE	7B	6D	28	:63
90	AD	A2	66	7F	16	75	F5	7F	:33
170	64	92	61	90	D8	D6	D7	00	:6C
250	40	D4	70	88	E0	00	2A	92	:A8
330	70	3A	88	E0	00	00	4E	BF	:1F
410	BB	BE	BE	BB	2B	00	CF	C2	:AE
490	C9	13	88	B0	4E	B8	76	B1	:41
570	AE	A6	9D	AB	9D	AD	2F	B5	:CA
650	E9	CB	BF	00	9E	EF	66	9F	:05
730	96	00	A2	AF	AD	B9	C2	B6	:C5
	72	FD	BD	AE	DD	D3	4D	75	/4C

LINE	+0	+1	+2	+3	+4	+5	+6	+7	SUM
810	C3	00	AB	C9	B1	A4	B4	3A	:7A
890	0C	D4	C0	BC	00	D2	B8	DA	:C0
970	E5	79	A2	A1	AA	8D	9C	A4	:18
1050	A4	90	94	98	95	A1	B5	AF	:FA
1130	B7	B1	BD	A5	9C	B5	C4	AA	:89
1210	86	3B	BF	97	4A	BA	AE	B0	:79
1290	C0	BE	BE	00	28	FE	BD	21	:40
	55	87	DB	FA	FE	11	EC	E2	/8E

SOFT REVIEW

●今月の評価委員会　狩野 茅　三枝 和子　佐藤 進一　杉田 正夫　竹田 紀子

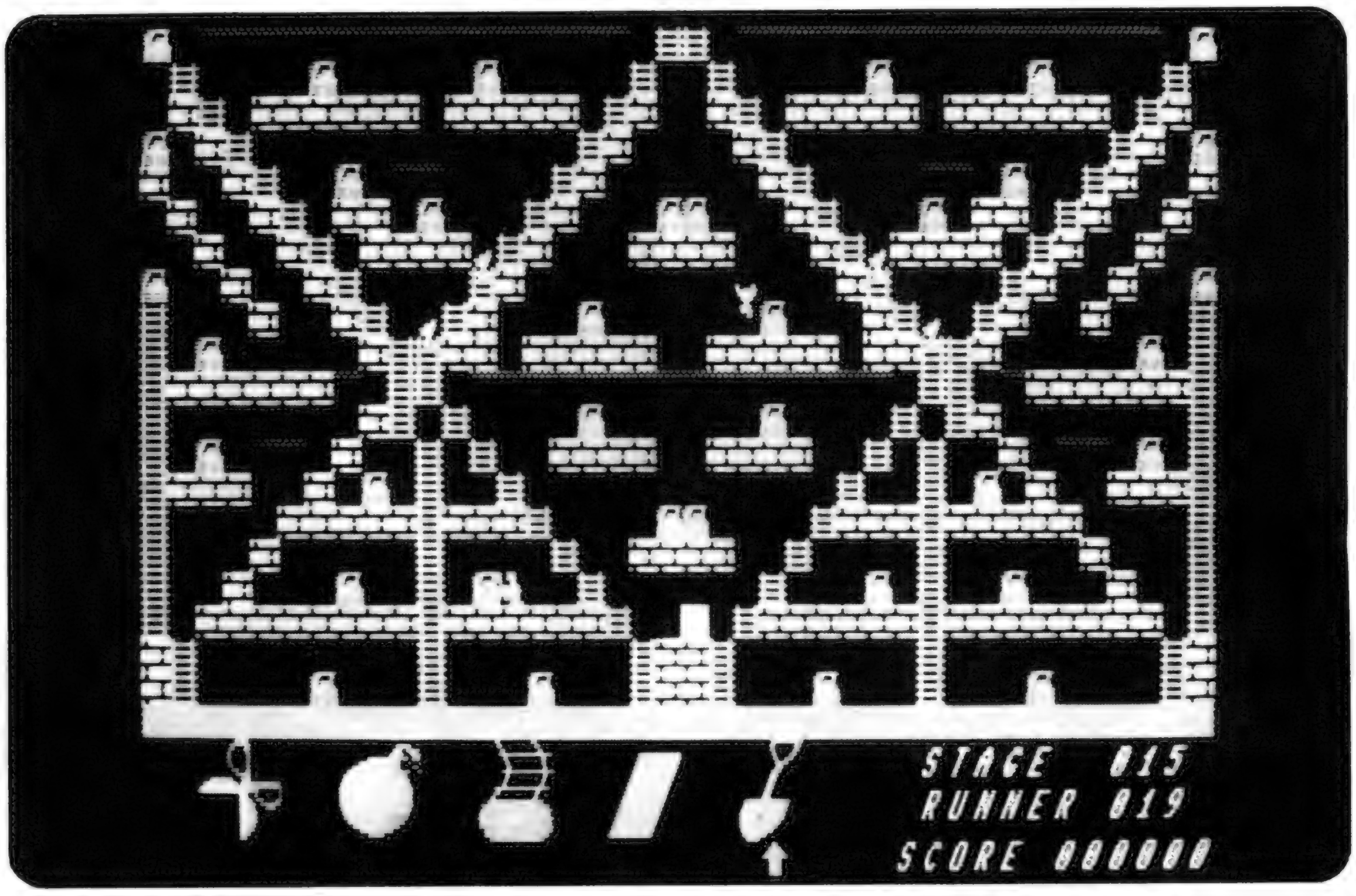
①ジャンル　②提供媒体〔T：カセット，3.5D：3.5インチ両面倍密度ディスケット(77のみ)，5D：5インチ両面倍密度ディスケット，5HD：5インチ両面高密度倍トラックディスケット(BS，AD2のみ)，8D：8インチ両面倍密度ディスケット〕　③価格　④適合機種（一部ソフトを除きとくに明記していませんがNEW7は7用のソフトすべて，77は7用のテープで供給されているソフトが動きます）⑤使用言語　⑥必要なオプション　⑦発売元

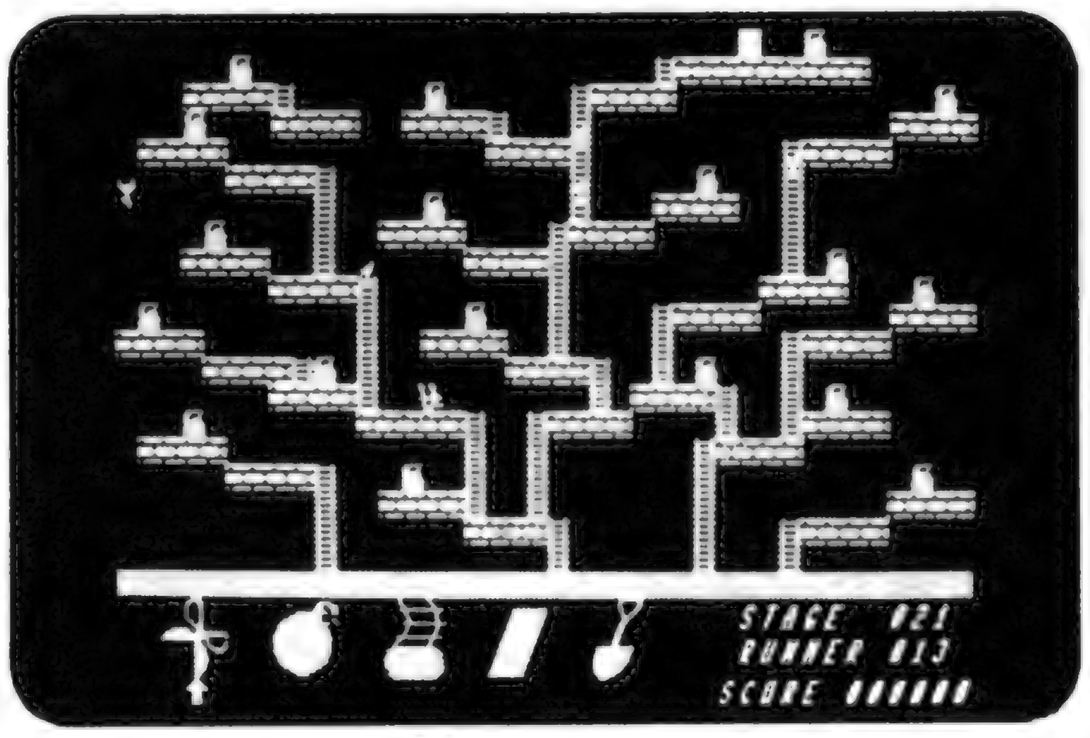
スタコラCRUSH

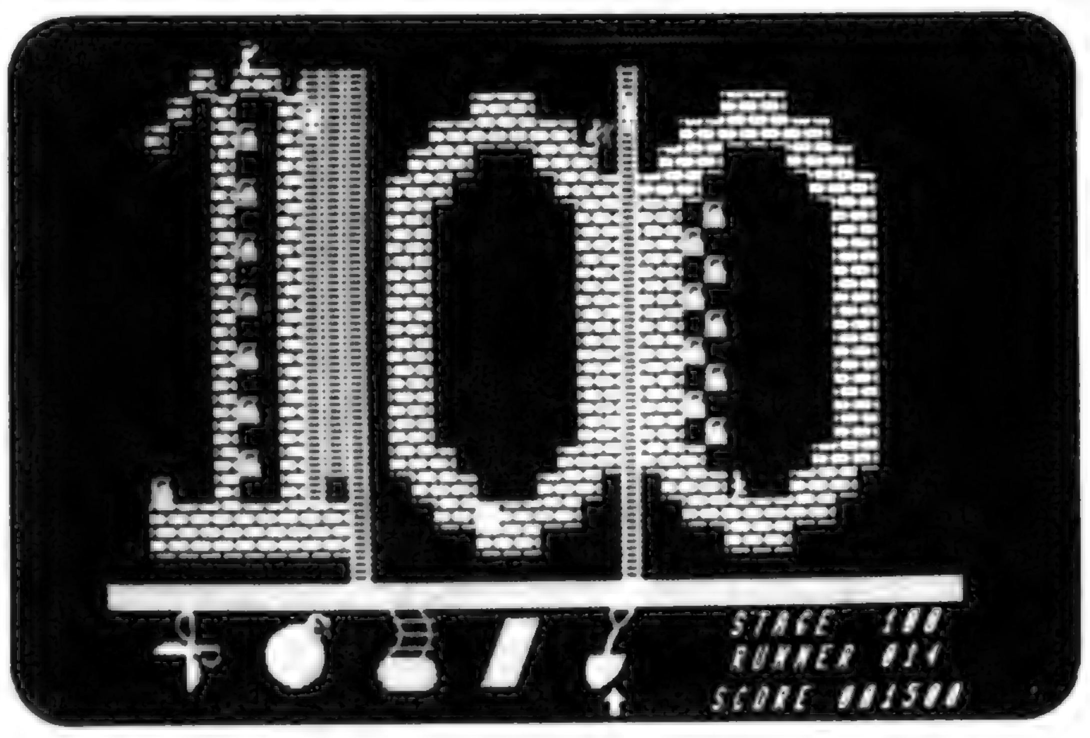
①思考型＋アクションゲーム　②T，5D
③T：4,800円　5D：6,800円　④7
⑤マシン語　⑥なし　⑦ジャコム

一見ロードランナー風だけど，道具を使う楽しみが加わっている。プレイヤはハサミ，爆弾，はしごなどを使いながら宝を奪う。反射神経のほかに頭脳が必要。全部で200面（テープ版は30面），ディスク版のみジェネレート機能つき。

	K.S.	M.S.
操作性	☺☺☺☺☺	☺☺☺☺
グラフィックス	☺☺☺	☺☺☺
効果音	☺☺☺	☺☺☺☺
スピード	☺☺☺☺☺	☺☺☺☺
オリジナリティ	☺☺☺	☺☺☺
熱中度	☺☺☺☺☺	☺☺☺☺☺
コストパフォーマンス	☺☺☺☺☺	☺☺☺☺







▶初めてこのゲームをやったとき，凝りすぎだな，って感じがした。だけど2回，3回とやって道具の使い方に慣れてくると，「ロードランナーより面白い！」と実感。どの道具をどこでどういうふうに使うかはプレイヤの意志。使わないのも自由。きわめて自由度の高いゲームだ。たとえば空間に自分ではしごをかけて道を作る，この"自分で～する"っていうのが面白くて，収められているパターンで遊んでいても自分で新たなパターンを作っている感覚になる。この快感は一度味わったらやみつきになりそう。それにスピードが速い。スピードはこのテのゲームの命だ。自分で調節できるのもうれしい（ランナーの人数もステージもすべて調節できます)。キーの反応もいいし，とにかく面白い!!　これはもう表彰したい。この夏，最高のソフトだ。(K. S.)

▶一目みて「ロードランナーの真似だ」と思い，遊んでみて「ロードランナーよりすごい」と感じた。これかなりのおすすめもの。文句なしにハナマル進呈。もうロードランナーはいりません。ほしくなんかないもんね。これは決して負け惜しみなんかじゃないからね。ロードランナーよりすごいのは自分ではしごを自由にかけられる，爆弾で穴をあけてその穴をふさぐ橋まである。穴を掘ったら自動的に埋まるなんて理不尽なこともない。これだけ使える道具が増えたんですから，パズルとしての楽しみが増えるわけ。音もより多彩になっています。えっ，ロードランナーにはコンストラクションもついてるって？　ちゃんとコレにもついてるんだ。とにかく面白いんだ。面白いんだけどね。ロードランナーと比較しなきゃ説明できないところがくやしい。パッと見たらそっくりだもんね。次回はコレと同じくらいゲーム性のある完全オリジナルゲームを作ってくださいまし。　(M. S.)

リクエスト受け付けます！

評価委員会ではコーナー充実を図るべく読者の皆さまから評価してほしいソフトのリクエストを受け付けます。官製ハガキに評価委員会へのご意見・ご希望をお書きのうえ，ソフト名と発売元を明記して送ってください。あて先は，

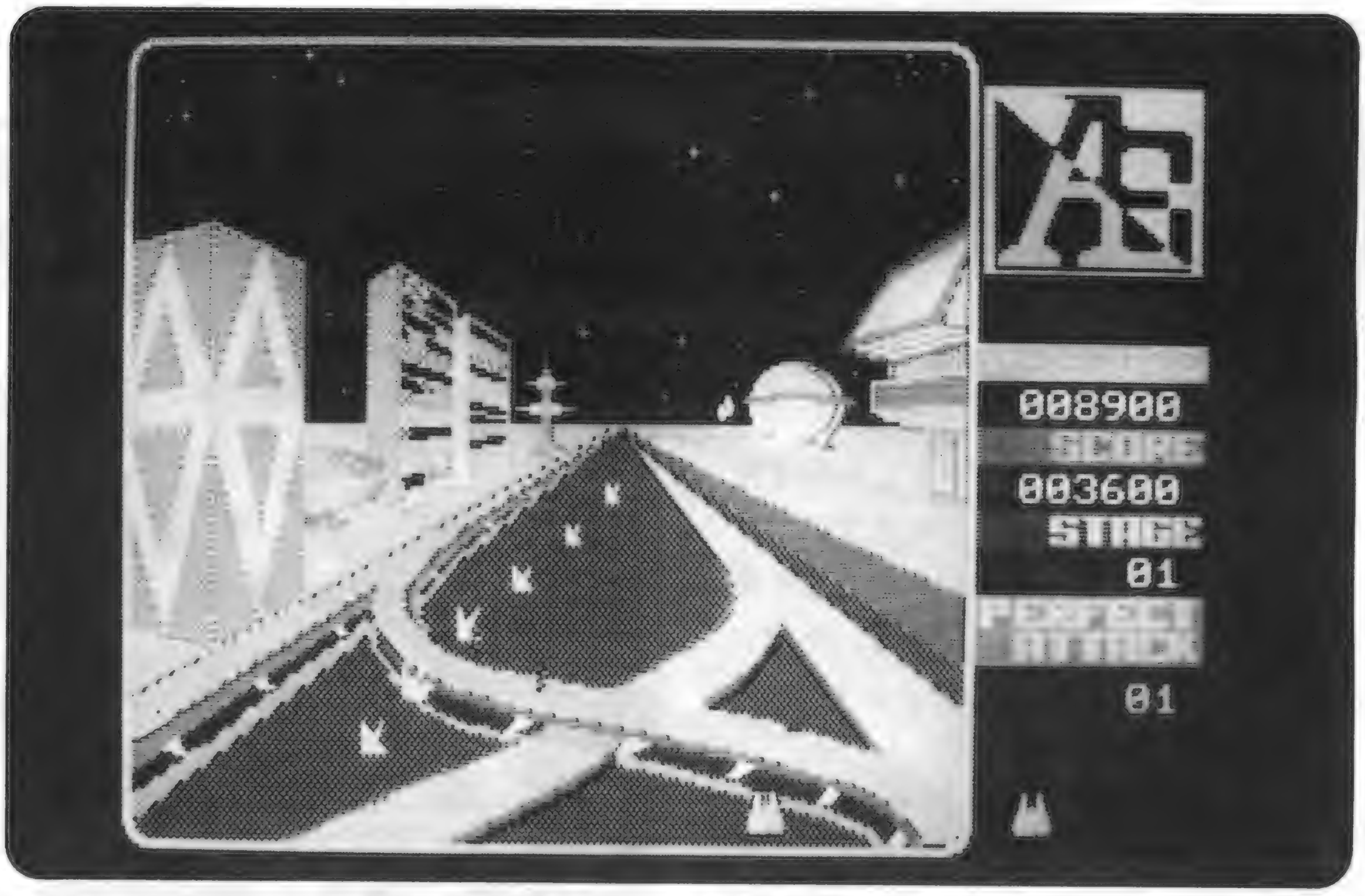
〒102　千代田区四番町2－1
㈱日本ソフトバンクFM編集室内
ソフトレビュー評価委員会

A. E.

①アクションゲーム ②3.5D，5D
③6,800円 ④7,8,NEW7,77 ⑤マシン語
⑥なし ⑦小西六エニックス

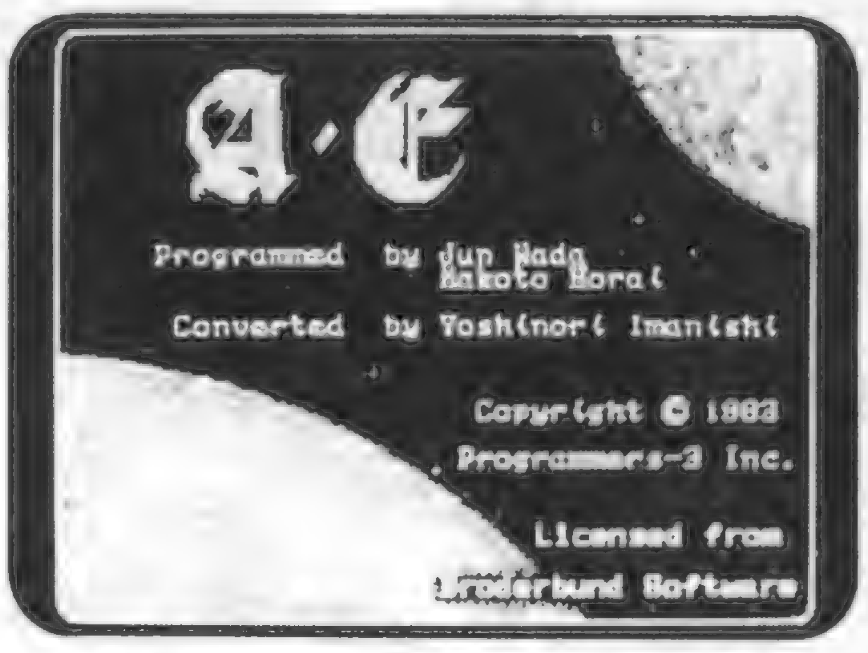
舞台は2450年テラ。コンピュータミスであばれまわるエイ型ロボット A.E. を退治する。アップルの移植版。各画面ごとにかわるテーマ音楽，グラフィックスがすぐれもの。全部で8画面。

	S.S.	K.S.
操作性	☺☺☺	☺☺☺
グラフィックス	☺☺☺☺☺	☺☺☺☺☺
効果音	☺☺☺	☺☺☺☺☺
スピード	☺☺☺	☺☺
オリジナリティ	☺☺☺☺	☺☺☺
熱中度	☺☺☺☺	☺☺☺☺☺
コストハフォーマンス	☺☺☺☺	☺☺☺☺☺



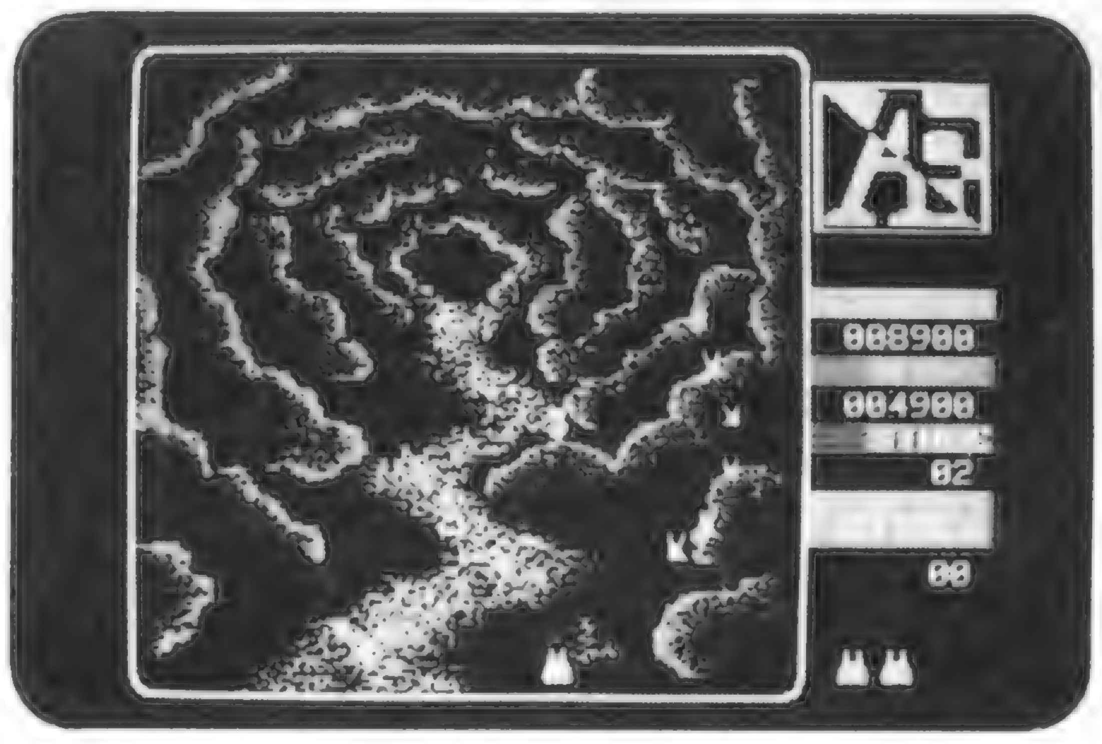
▶あの〝A.E.〟の FM 版がついに登場しました！ グラフィックスの美しいこのゲーム，アップルで DEMO っているのを憧れのまなざしで眺めていた人もさぞ多いことでしょう。しかし，今日からはあなたの FM で A.E. が走るのです！ だけど……FM シリーズの宿命でしょうか，やはりキー操作がよくありません。 強敵 A.E. をむこうにまわして，このハンデはちょっと痛い気もしますが……。ここでゲームのやり方について補足を。① A.E. を3回パーフェクトで全滅させると次の面に進みます。②宇宙船は2万点で1機追加されます。③5面以降は A.E.が2列，3列に分かれて攻撃してくるようになるので，次の面に進むにはテクニックを要します。なお，A.E. が落とす弾はかなり見づらいので注意してください。

(S. S.)



▶待ちに待った A.E.，ついに FM 版が出た。移植版というとすぐ本家と比べてしまうけど，FM 版 A.E. はまずまずのでき。キー操作のまどろっこしさには閉口するけど（これは FM の宿命と思ってあきらめよう），とにかくグラフィックスが見事だ。効果音も最高。もう5重マルあげる。A.E. の飛来のしかたもいい。弧を描くように飛んでくる。だからこそこちらの攻撃も思うようにならないし，そこがこのゲームを面白くしている。画面数は8面までしかないけれど，3面ぐらいから難しくなってくる。8面までいったらたいしたもの。グラフィックスと効果音，ゲーム自体の面白さ，エンタテイメントなソフトだ。

(K. S.)



花札狂

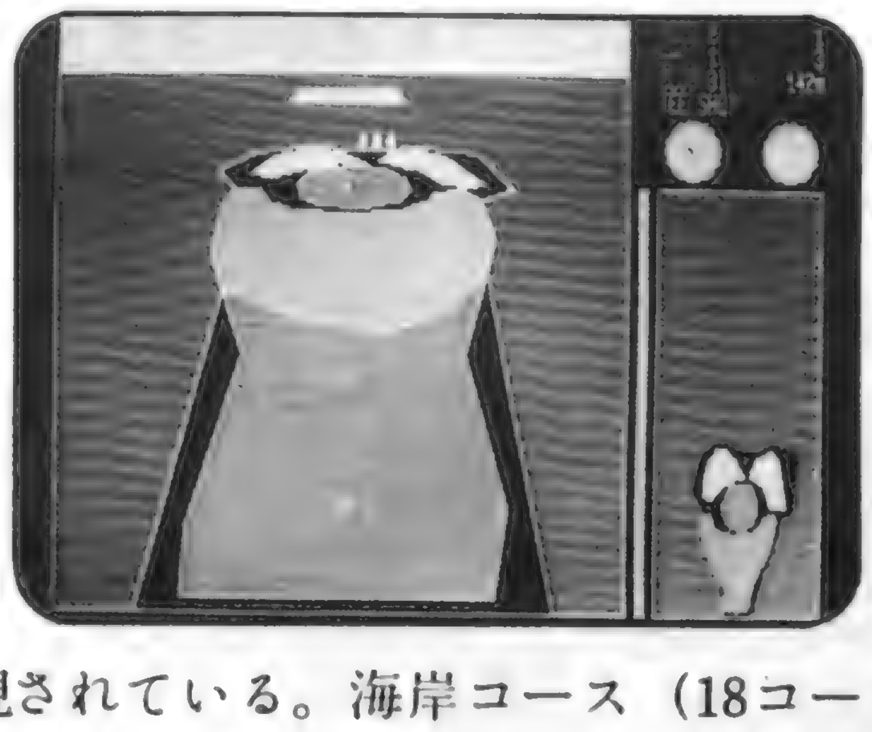
①ホビー ②T ③4,000円 ④7 ⑤マシン語 ⑥なし ⑦ハドソン

コイコイ，ばかっ花，おいちょカブの3種を収録。

3Dゴルフシミュレーション Part2

①ホビー ②T ③4,000円 ④7,NEW7,77 ⑤マシン語 ⑥なし ⑦T&Eソフト

視野画面，ボールの軌跡などが3次元で表現されている。海岸コース（18コース）でパー72。ボールが見づらいのが難点。

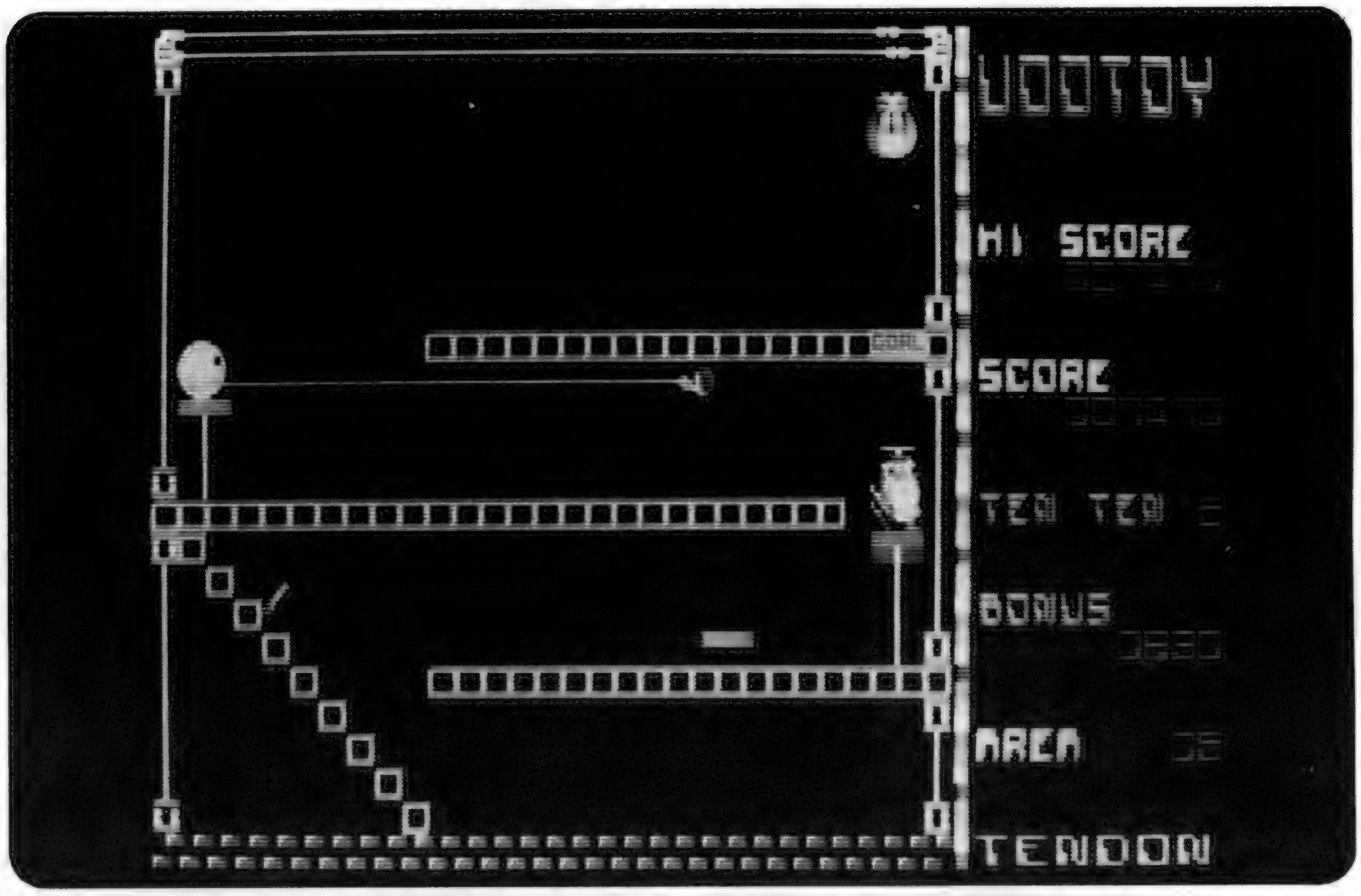


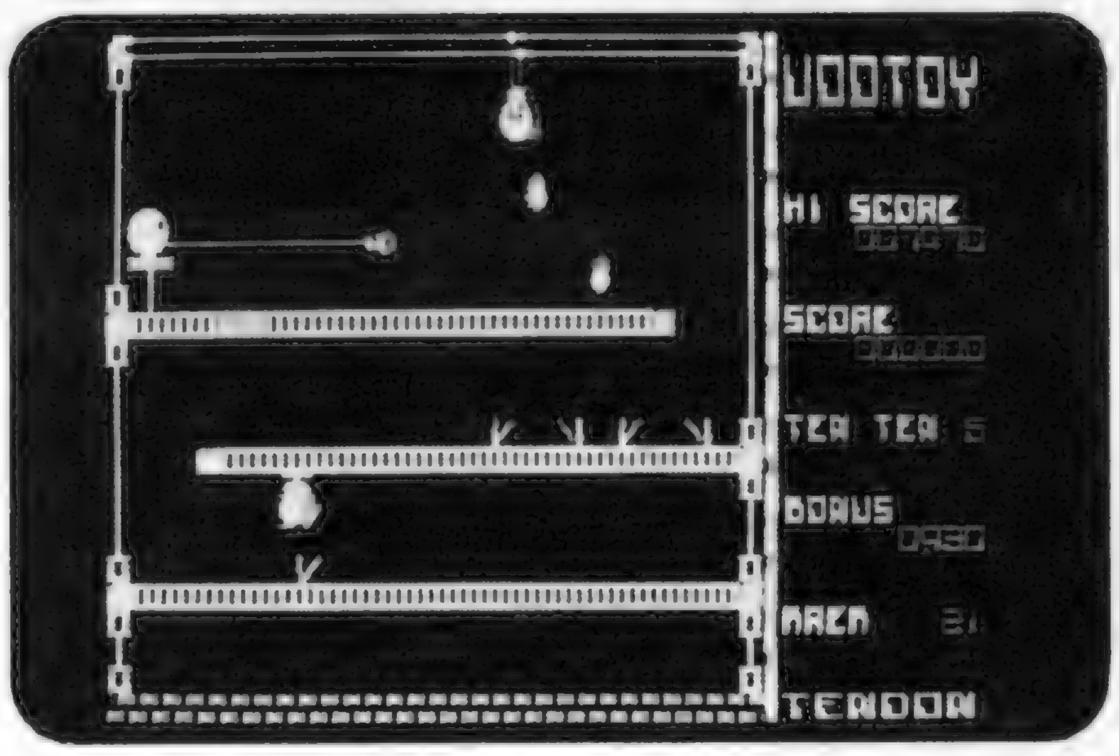
ウットイ

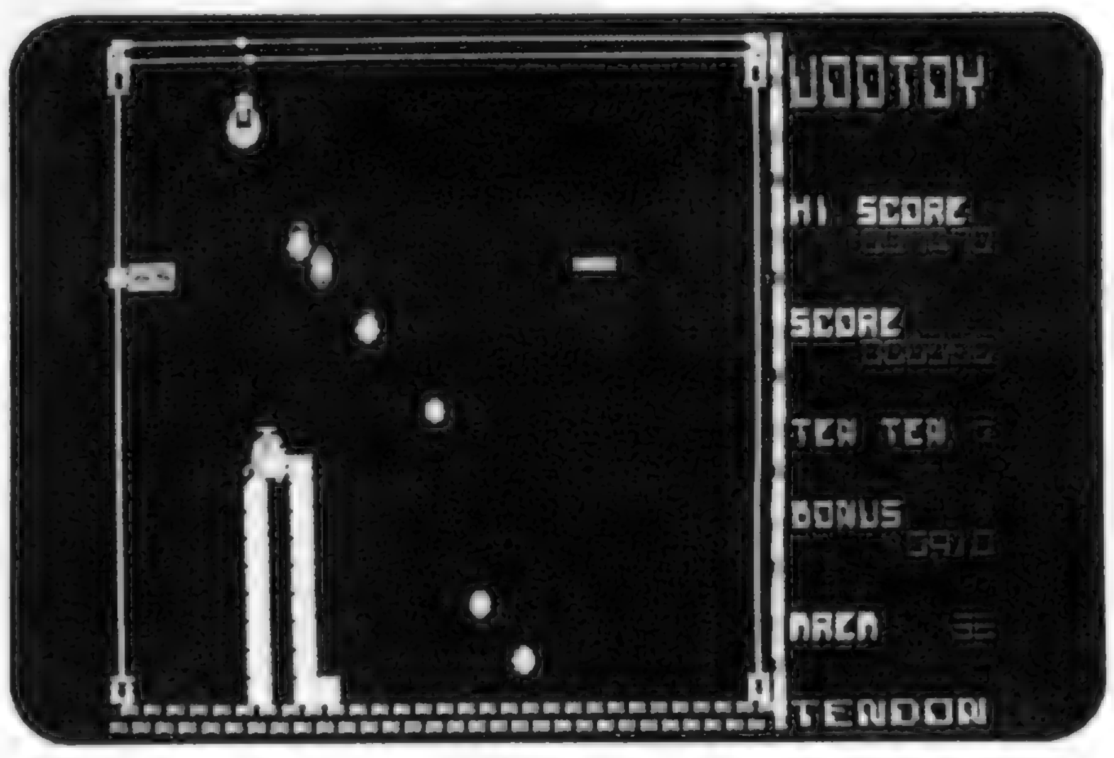
①思考型＋アクションゲーム ②T,5D
③T：4,800円 5D：5,800円 ④7,NEW7,77 ⑤BASIC＋マシン語 ⑥なし
⑦コムパック

I/O 8月号掲載のバージョンアップ版。マッチ棒，トランポリン，エレベータなどの脇役をうまく利用してテンテンをゴールインさせる。画面数は100面。コンストラクションつき。

	K.K.	N.T.
操作性	☺☺☺	☺☺☺
グラフィックス	☺☺☺☺	☺☺☺☺
効果音	☺☺☺	☺☺☺
スピード	☺☺☺	☺☺☺
オリジナリティ	☺☺☺☺☺	☺☺☺☺☺
熱中度	☺☺☺☺☺	☺☺☺☺☺
コストパフォーマンス	☺☺☺☺	☺☺☺







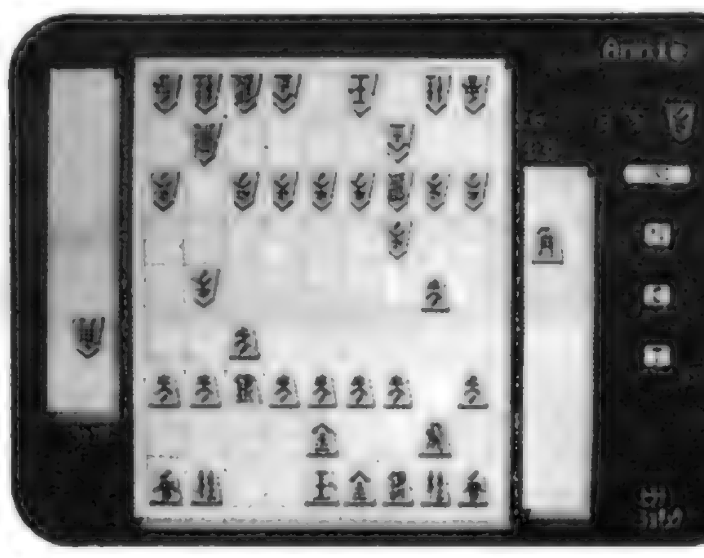
▶わあーうっとーしー，なんてダジャレを言ってる場合ではなくて……，マッチ棒が不気味ですねぇ。動きがイヤらしいし，触れたら死んじゃうなんてくやしい。マッチがたくさん出てくるともう大変！ タイミングよくよけないとアウト。それと雨あられのように爆弾を落とすランプ。これも難物，あまり近付かない方がベター。ベロをうんべろべ〜って出す奴は面白い。上に乗れるし下からドックと縮んでいく。上にいるときは氷が落とせるし，面白い。空中で方向を変えるのが1つのテクニック。それと氷を積むときESCキーを押してからSTOPキーを押すのがコツ。どんどん積めるからネ。それから……，これ以上は教えない方がいいかな？ オリジナルな感じやグラフィックスはよい。音が当たり前すぎる点が残念。100面もあると飽きがこなくていいネ。コンストラクションも付いていて結構長く楽しめそう。 (K. K.)

▶ムフフッ！ とても白熱しちゃった。キャラクタもかわいい，凝っているしいいね。今までのいろいろなゲームをミックスして練ったようなゲーム。スタートからゴールまでどう工夫してたどり着くか，というゲームなんだけど，爆弾は落ちてくるし，針の道はあるし，パンチは飛んでくるし，もう大変。でもうまく氷を積んだりこわしたり，動く絨毯(じゅうたん)に乗って移動したりしてゴールにたどり着く。リアルタイムだからタイミングを考えて頭をFullに働かせないとダメ。あっ，あと感動したのはジャンプしながら方向転換できるんだよね。100面あってそれぞれにいろんな工夫がなされている。どうにか頑張ろう。なかなか面白いと思うから，私としてはオススメ品です。(N.T.)

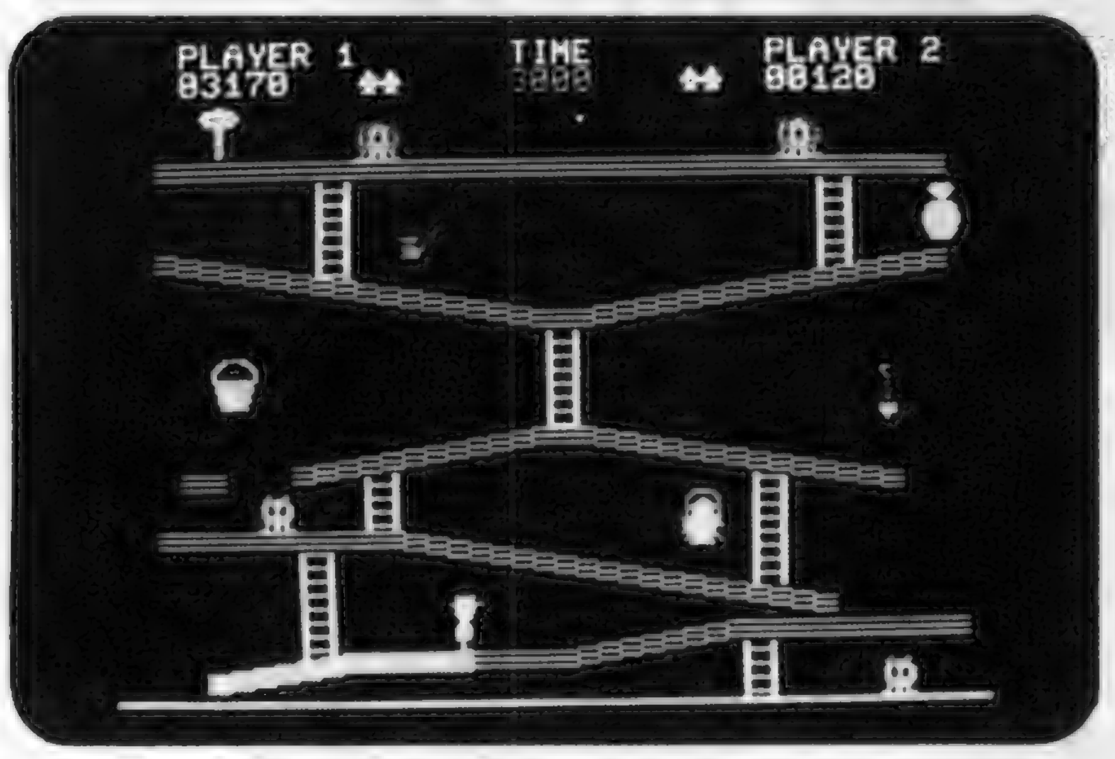
名将

①ホビー ②T,3.5D,5D ③T：3,800円 3.5D•5D：5,800円 ④7,NEW7,77
⑤BASIC＋マシン語
⑥(シリアル回線使用の際のみ)RS-232C
⑦日本ソフト&ハード社

対局と詰め将棋のセット。対局では平手，2枚落，4枚落のいずれかを選択でき，詰め将棋ではコンピュータの出題のほかに自分で問題を作成できる。ディスク版ではシリアル回線を使って，遠く離れた人と対局可能。コンピュータ側の強さが選べる（3段階）し，思考時間も短い。ただ，グラフィックスにひと工夫ほしかった。



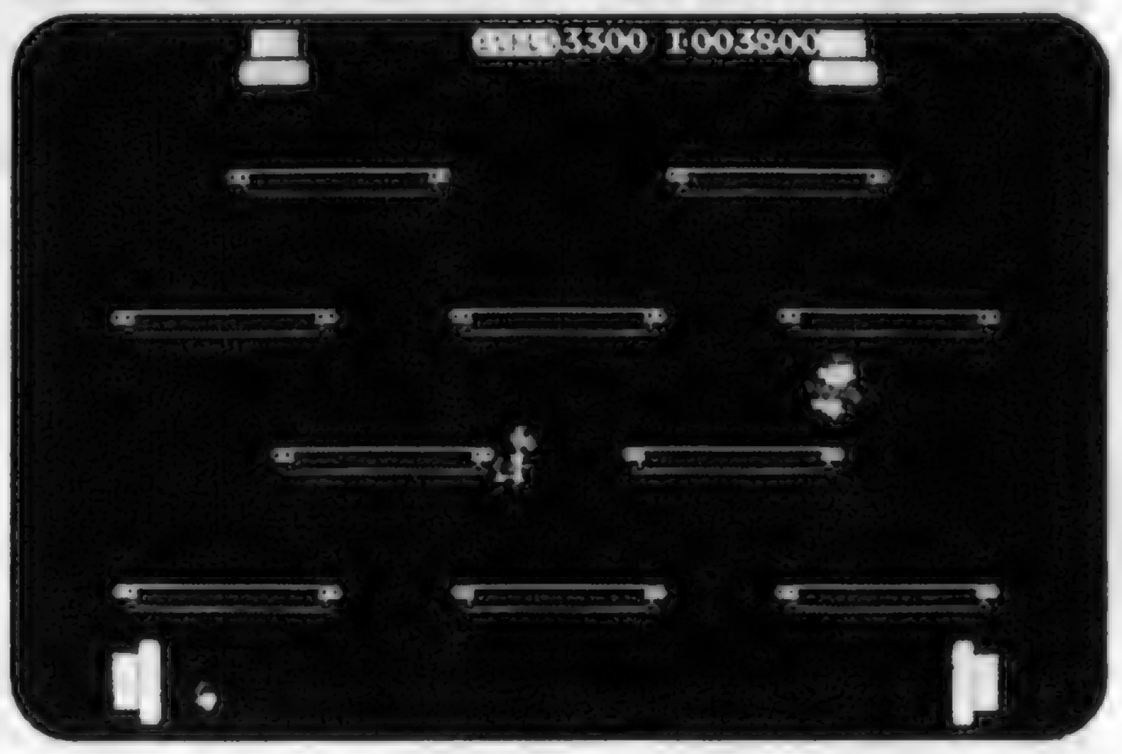
MINER 2049er



①アクションゲーム ②3.5D, 5D
③6,800円 ④7,8,NEW7,77 ⑤マシン語
⑥なし ⑦小西六エニックス

アタリの移植板。ウラニウム炭鉱に逃げこんだ殺人犯を追跡するが，ミュータントが邪魔をする。主人公のバウンティ・ボブをコントロールして，殺人犯を捕まえろ。

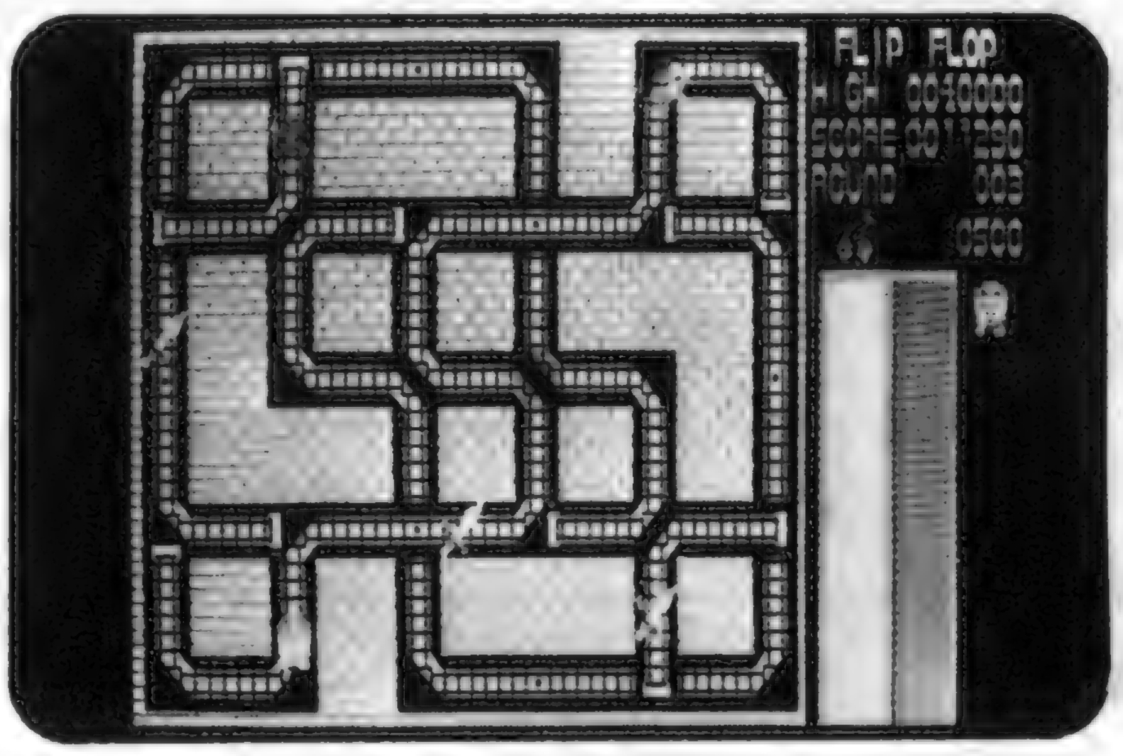
マリオブラザーズ スペシャル



①アクションゲーム ②T ③3,600円
④7 ⑤マシン語 ⑥なし ⑦ハドソン

ゲームセンターの本家と比べちゃうとがっかりするけど，そういうことを忘れて遊べば面白い。スピードもまともだし，キャラクタもかわいい。でもどちらかといえば，入門者向け。2人でも遊べる。

フリップフロップ



①思考型＋リアルタイムゲーム ②T ③

イージーアクセス

①データベース ②3.5D, 5D(マニュアル付) ③25,000円 ④7,8,77,11 ⑤BASIC ⑥プリンタ ⑦ケンブリア・ランゲージ・サービス

英文文献検索ソフト。著者名やキーワードによる検索が行える。複数項目による検索も可能。検索結果をソートし，任意の書式に変換できるリファレンス機能も持つ。さらに変換されたファイルは「イージーライター・スーパーVer.II」で編集・印字できる。

イージーライター・スーパーVer.II

①英文ワープロ ②3.5D, 5D(マニュアル付) ③20,000円 ④7,8,77,11(BSを除く) ⑤BASIC＋マシン語 ⑥プリンタ(電子タイプライタも可) ⑦ケンブリア・ランゲージ・サービス

アンダーライン，強調指定，化学式の印字が可能。行末処理は自動的に行う。Ver. I をさらに機能アップした。

WP-1

①ワープロ・作図 ②5D(マニュアル付) ③38,000円④7,8⑤BASIC⑥漢字ROM，フロッピ2ドライブ，プリンタ(MP-80タイプIIIおよび相当品) ⑦日本ソフトウェア開発

ワープロに作図機能を統合化したもの。入力はカナ漢字変換で，JIS第一水準をサポート。40文字までの文字列登録ができる。線，箱，円，弧が描け，削除も可能。また作成した文章やグラフィックスをBASICのステートメントに自動変換するので，プログラムの作成が容易。

給与計算

①給与計算 ②5D(マニュアル付) ③150,000円 ④7,8,NEW7,EX ⑤BASIC ⑥フロッピ2ドライブ，プリンタ ⑦京葉システム

月例給与計算，賞与計算，社会保険算定から年末調整までの一連の処理を行う。給与明細表，銀行払込表，金種表などをプリントアウト。対象人員は最高150人。

The 葉書

①ビジネス ②5D(マニュアル付) ③38,000円 ④7 ⑤マシン語 ⑥漢字ROM，フロッピ2ドライブ，プリンタ(MZ-1904, MP-80, RP-80, GP-700) ⑦ジャコム

暑中見舞，年賀状などを絵と文章を組み合わせてハガキ印刷するソフト。定型的な文章は用意されているが，自分で作成することも可能。また，筆文字(32種)やさし絵(47種25キャラクタ)もあらかじめシステムに組み込まれているので，即使用できる。

英文法学習

①CAI ②3.5D, 5D(マニュアル付) ③中1,中2,中3用：各6,800円 受験用：7,800円 ④7,77 ⑤BASIC＋マシン語 ⑥漢字ROM ⑦一夢堂

各学年の英文法を「重要事項のまとめ」「練習問題」「テスト」の3段階に分けて学習する。ウインドウ表示，マルチページ機能，サウンド機能を生かして見やすい表示となっている。文法事項は選択できるので進度・やる気に合わせて学習できる。

FM-積算PARTIII 内装

①科学技術 ②5HD(マニュアル付) ③400,000円 ④BS ⑤BASIC ⑥漢字プリンタ ⑦ケンブリア・ランゲージ・サービス

登録済みの一覧表から下地材，仕上材を選択し，部屋ごとに素材のコードを分類し

3,600円　④7,NEW7,77　⑤マシン語＋BASIC　⑥なし　⑦エニックス

魔法使いに大事な〝魔法の剣〟を盗まれてしまったデビル君。何とか剣を取り戻さないと一人前の悪魔になれない。交差点の向きを変えてうまく剣を取り戻そう。のんびり考えている暇がない思考型。キャラクタが可愛い。

暗黒城

①アドベンチャゲーム　②T　③3,800円　④7,NEW7,77　⑤マシン語＋BASIC　⑥なし　⑦エニックス

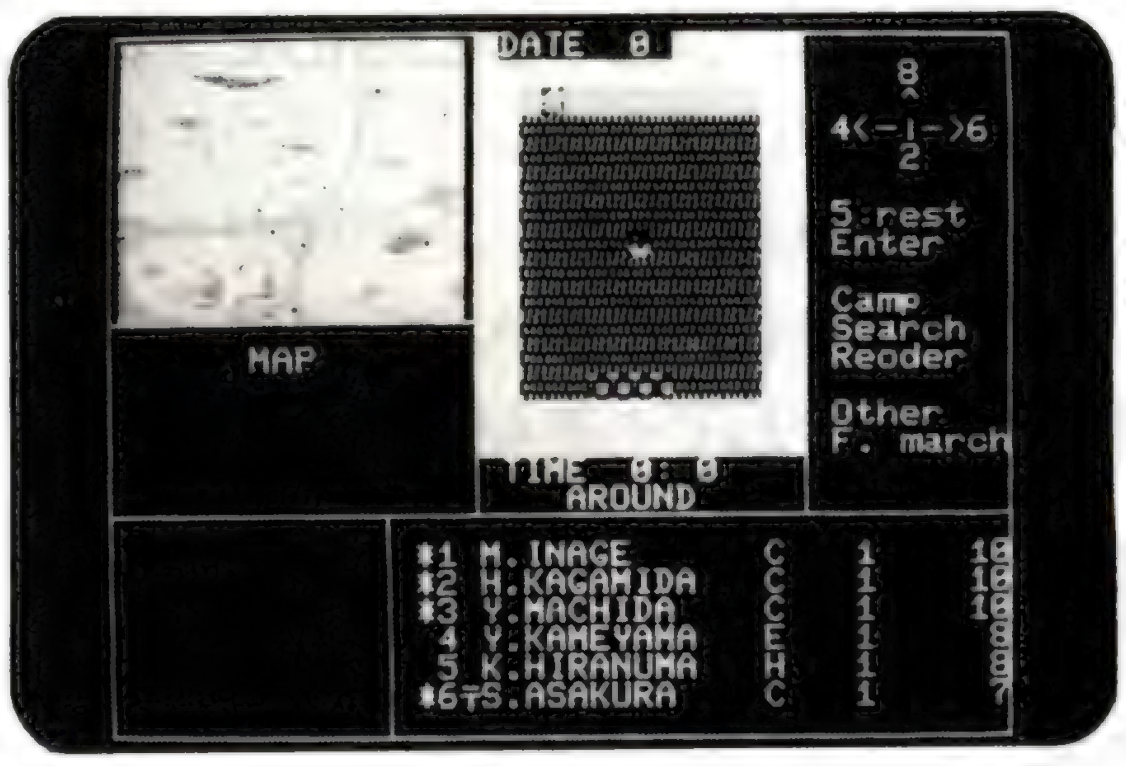
西暦2188年，プレイヤはリュウ・エンジェル博士となりミネル・バトンへの旅に出る。

そしてラパルの鏡を割って，創造神ギャリアンの理想郷にたどり着かなければならない。

フロントコマンド

①シミュレーションゲーム　②5D　③6,800円　④7　⑤BASIC＋マシン語　⑥なし　⑦ラポート

6人編成の小隊を率いて命令を遂行していくシミュレーションウォーゲーム。兵員は戦闘経験を積むごとに能力が向上し，強力な部隊となっていく。



ていく。プリンタ出力は漢字表示で部屋別，素材別に集計し，計算式もプリントアウトする。

パソコン構造力学

①科学技術　②5D(マニュアル市販図書)　③13,000円　④7,8,EX　⑤BASIC　⑥漢字ROM，フロッピ2ドライブ，プリンタ(MB27402・27405・27406)　⑦アーキソフト社

宮澤健二著「パソコン構造力学――力学挙動の可視化――」(啓学出版)をグラフィックシミュレーション化。処理内容は連立方程式，固有値問題，整形ラーメンの総合問題など9種21個。必要に応じプリントアウト，ハードコピーが取れる。構造技術者向けだが，専門外の人も楽しめる内容となっている。

建築構造のパソコングラフィックス

①科学技術　②5D(マニュアル市販図書)　③23,000円　④7,8,EX　⑤BASIC　⑥漢字ROM，フロッピ2ドライブ，プリンタ(MB27402・27405・27406)　⑦アーキソフト社

宮澤健二著「建築構造のパソコングラフィックス――力学挙動の可視化――」(啓学出版)のグラフィックシミュレーション版。

OS-9 Utility Command Vol.1

①ユーティリティ ②5D,5HD,8D(マニュアル付)　③5D：12,800円 5HD・8D：14,800円　④7,8,11(BSを除く)　⑤C　⑥OS-9　⑦星光電子

OS-9のファイルを扱ううえで役立つ便利なユーティリティコマンドをパッケージにしたもの。ソフト開発の効率を高めるばかりでなく，OS-9の強力な機能のひとつであるパイプラインの使い方を学ぶのにも，大いに役立つ。コマンドは以下の5つ。

▶Dlist◀DirとListコマンドの2つの機能をまとめたもの。指定されたファイルがディレクトリファイルのときはファイル名を1行ずつ表示し，普通のファイルのときはListコマンドと同じようにそのファイルの内容を表示する。

▶Wildc◀標準入力パスからテキスト行を読み込み，指定したパターンと一致する(しない)行を標準出力パスに出力。

＊(あらゆる文字と一致)，?(任意の1文字と一致)，〔 〕(指定された文字のうち任意の1文字に一致)，￥(＊や？などの特殊記号の取り消し，文字のコード指定)の記号でパターン指定できる。

例：BASIC09のプログラムからRUN文を含む行を抜き取る。

```
Wildc<proc1   *RUN*
```

Cのソースファイルだけをリストアップ

```
Dlist ! Wildc *.c
```

▶Lgen◀標準入力から読み込んだテキスト行を，文字，単語，行単位で，引数で指定したレイアウトで，1文字変数(?)，単語変数(＊)，行変数(−)に代入して，1行ずつ標準出力パスに出力する。

例：現在のディスクトリ内のバックアップファイルだけを削除する

```
Dlist ! Wildc *_bak ! Lgen
Del * ! shell -p t
```

▶BootLink◀任意のファイルを，そのファイルの存在するディスクのセクタ0にリンクする。システム起動時にメモリにロードするファイルを自由に変更できるため，頻繁にシステム構成を変更する場合に便利。

▶Owner◀ファイルのオーナIDを変更する。友人などからファイルをもらった場合，その友人のオーナIDが付いていて使いづらい場合があるが，このコマンドでオーナIDを変更すればよい。

for FM-7, 8

F-BASICプログラムリストを生成

グラフィックプログラムジェネレータ

吉田 昌

オールマシン語のグラフィックツールを作ってみました。作りあげたデータは，なんとF-BASICのテキストそのもの。つまり，これを使って絵を描いていくだけで，即，BASICのプログラムができてしまうという実に楽しいプログラムなのです。グラフィックのプログラムは，コマンドやデータの打ち込みが，やたらと時間がかかるものです。しかし，これさえあれば，相当複雑なプログラムでも楽々と作れます。

使用前に

＄6000から＄66CDに置かれているこのプログラムは，ポジションインディペンデントのマシン語プログラムです。リストのとおりに打ち込んだら，(FM-8の人は修正部分を直して，) SAVEM"G.P.G." , & H6000, & H66B4, & H6000でセーブします。このプログラムは，BASIC インタプリタの拡張HOOKを使っているので，バグがあったときはたいへん危険です。打ち込み終えたら，EXECする前に必ずセーブしましょう。ディスクの方は，ディスケットを出してから，走らせます。デバッグが完全に済むまでは，この作業を必ず繰り返してください。

使用法

まず，BASIC のプログラムがなにもない状態にしてください。これは「NEW⏎」と打ちます。次に，「CLEAR, &H5FFF」と打って，スタックエリアを移します。現在のプログラムに，グラフィックプログラムを付け足したいときは，NEW の必要はありません。

「EXEC &H6000⏎」で，プログラムは起動します。プログラムが起動すると，まずBASIC のプログラムを実行します。そして，暗黙 END になったとき（END やSTOP命令のないプログラムが終了したとき）に，Ready状態に戻らず，メニュ状態となって，0行目に，コマンドメニュが表示されます。

各コマンドの説明

クリアコマンド（クリアキー）

CLEAR または CLS キーを押すと，「CLS 0」が生成されて，画面がクリアされます。プログラムの最初に使います。プログラムそのものはクリアされませんから注意してください。

ラインコマンド（Lキー）

メニュ状態で，L キーを押すと，グラフィックカーソル(+)が表れます。これを，線の引き始めに移動させて（カーソル移動キー ←↑↓→ および SHIFT ＋カーソル移動キーを使う），⏎キー(FM-8はReturnキー）を打ち，次に線の引き終わりに移動させて再び⏎キーを打つと，画面上に線が引かれます。この時点で，すでにLINE文が1つ生成(ジェネレート)されていますが，続けて線を引くときには，さらにカーソルを移動させて，⏎キーを打つという作業を繰り返します。

ラインコマンドを終えるときは，カーソルを移動させずに，⏎キーだけ打つと，メニュ状態に戻ります。

ここで，E キーを押して，LIST をとってみると，あら不思議，今引いた線がそのままLINE文となって出てきます。もちろん "RUN" すれば，絵を再生できますし，普通に打ち込んだプログラムとなんら変わるところはありません。

カラーコマンド（0-7キー）

0～7のテンキーを押すと，そのパレットコードによるグラフィック命令を生成するようになります。また，グラフィックカーソルの色も，これと同じになります。

ファンクションコマンド（Fキー）

F キーを打つと，0行目に機能が表示されますから，0～5キーでこれを選択します。以後は，この機能によるグラフィック命令を生成します。

NOT機能はSYMBOL文以外は不要なので，NOTを選択しても，他の命令では，XORを生成しますから，注意してください。

ドットコマンド（Dキー）

Dキーを打つと，PSETとPRESET を生成します。機能が PRESET のときに，PRESET を生成して，機能が PSET のときは，AND と ORで，PSET を作っています。

このコマンドも，メニュに戻るには，グラフィックカーソルを動かさずに⏎キーを押してください。

ペイントコマンド（Pキー）

メニュ状態で P キーを打つと，PAINT文を作ることができます。グラフィックカーソルを，塗り始めの位置にもってきて，⏎キーを打つと，0行目で，パレットコードを聞いてきますから，0～7キーを打っていくと，その値が次々と表示されます。間違えて打ったときは，⌫(バックスペース) キーで修正できます。最後に⏎キーを打つと，ペイントを実行して，メニュに戻ります。

REMコマンド（Rキー）

メニュ状態で，R キーを打つと，REM文（ ' 文）が生成できます。長いプログラムになると，現在どこのプログラムを生成しているのかがわからなくなるので，これを使って，注釈をつけておくと便利です。

0行目に「REM.?」のメッセージが表示されたら，注釈文字列を打ち込んで，⏎キーでメニュに戻ります。←キーで修正もできます。

シンボルコマンド（Sキー）

メニュ状態で，S キーを打つと，シンボルコマンドを実行します。まず左上のドットの位置を決めて，次に表示するときの角度を決め，表示する文字列を入力します。そうすると，文字列が表示されるので，今度はカーソル移動キー(および SHIFT ＋カーソル移動キー）を使って，横と縦の倍率を決めます。→キーは横方向の拡大，←

キーは横方向の縮小，[↑]，[↓]キーは縦方向の拡大・縮小です。また，この状態で，[2]，[4]，[6]，[8]キーを使って，位置を移すこともできます。[8]，[2]キーは上下の移動，[4]，[6]キーは左右の移動です。[GRAPH]キーといっしょにこれらのキーを押すと，10ドット単位で移動します。[0]キーを押せば，角度も変わります。最後に↵キーを押すと，SYMBOL 文を実行してメニュに戻ります。

BOXコマンド（[B]，[b]キー）

[B]キーで，BOX FILL，(LINE文のBF)[b]キーでBOX 文（LINE 文のB）が生成されます。グラフィックカーソルで，左上と右下の位置を決めてください。

サークルコマンド（[C]，[c]キー）

メニュー状態で，[C]キーを打つと，CIRCLE文が生成できます。グラフィックカーソルを中心に移動させて↵キーを打つと，画面に円が表れます。[←]，[→]キーおよび[SHIFT]+[←]，[→]キーで円の大きさを決めて，[↑]，[↓]キーおよび[SHIFT]+[↑]，[↓]キーで円の比率を決められます。[2]，[4]，[6]，[8]キーで移動もできます。↵キーでCIRCLE文を実行して，メニュ状態に戻ります。円の大きさや，比率は，そのまま保存されます。大文字のときは“F”，小文字のときは“N”が生成されます。

ARC コマンド（[A]，[a]キー）

円の大きさと比率を決めるまでは，サークルコマンドと同じです。その後で，[→]，[←]，[SHIFT]+[→]，[SHIFT]+[←]キーを使って，円の開始位置を決めて，↵キーを打ちます。次に終了位置を決めて↵キーを打つと，CIRCLE文が生成されて，実行した後でメニュに戻ります。大文字のときは“F”，小文字のときは“N”が生成されます。

エンドコマンド（[E]キー）

プログラムの終了のときに使います。これを使うと，プログラムの開始時に拡張されたフックがもとに戻って，コマンド待ち状態（Ready 状態）に戻ります。

BREAK（[BREAK]キー）

実は，キー入力待ちのときに，[BREAK]キー（FM-8は[STOP]キー）を押しても，コマンド待ち状態に戻ります。しかし，このときは，BASICが拡張されたままなので，RUN すると，BASIC実行後にメニュ状態になってしまいます。「L..↵」で，最後に生成した行を表示できますし，「C.↵」で，メニュに戻ることもできるので，生成したプログラムの修正のときに使用すると便利です。

その他の注意

BASICプログラムを消さずに，EXECすると，まず BASIC のプログラムを，先に実行してから，メニュ状態になります。そして，前のプログラムに続けて，プログラムが生成されます。BASIC のプログラム中には，どんなコマンドがあってもかまいませんが，暗黙 END にならない限り，メニュ状態にはなりません。たとえば，END とか，DELETE などのように，実行後そのままコマンド待ちに戻ってくる命令があった場合や，エラーが生じたときはメニュ状態にはなりません。ただし BASIC は拡張されたままです。

メニュ状態で，“E”コマンドをするか，「POKE & H263，& H39↵」としない限り，拡張は解除されません。拡張したままマシン語プログラム本体をこわすようなことは，絶対に避けてください。

メッセージは，すべて SYMBOL のXORで書いてあるので，背景になにか書いてあると，判別できなくなることがありますので，あしからず。

メニュがいちいち表示されるのが，じゃまになったら，＄64FCの＄50を＄00に変えてください。

F-BASICの構成

プログラムテクニックの説明の前に，F-BASIC の構成を簡単に話しておきます（図1）。

マシン語ダンプの16バイト表示について

本誌掲載のダンプリストは，4月号に掲載の「使い勝手のよいマシン語モニタ」を使っています。本プログラムは走らせる前の状態がWIDTH 40のとき8バイト，WIDTH80のとき16バイト表示になります。また，WIDTH80のときにはオプションコマンドで8バイト表示も可能です。

また，本誌第3号（'83年6月1日発行）および「FILES Oh! FM」に掲載の「Oh! FM用チェックサム」プログラムを使って，横16バイト表示にさせたい方は，左のように各行を変更してください。この変更によって，横16バイトのチェックサムがとれます。なお，縦サムは表示されません。

●変更行のみ●

```
100 WIDTH 80,25:CONSOLE 4,21,0,0:COLOR 7
110 DIM CSUM(15,15)
160 PRINT 'Add   +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F  Sum';
200 FOR H=ADS TO ADE STEP 256
250 WIDTH 80,25:END
270 FOR H=ADS TO ADE STEP 256
320 WIDTH 80,25:END
410  FOR I=H TO H+255 STEP 16
450   FOR J=0 TO 15
470    SUM=SUM+PEEK(I+J):CSUM(J,(I-H)¥16)=PEEK(I+J)
830  LINE(DX*8,DY*8)-(DX*8+7,DY*8+7),XOR,4,BF
840  FOR TI=0 TO 100:NEXTTI
880 SUM=0:FOR S=0 TO 15:SUM=SUM+CSUM(S,DY-4):NEXT S:LOCATE 55,DY:PRINT RIGHT$('0
'+HEX$(SUM),2)
```

F-BASICは，通常の場合，コマンド入力ルーチンと，BASICインタプリタをメインルーチンとして作動しています。コマンド入力ルーチンでは，BASICテキストの作成およびダイレクトコマンドの実行が主な仕事です。インタプリタは，BASICテキストおよびダイレクトテキストの実行が主な仕事です。

BASICのテキストは，画面から入力されたコマンドが，一度中間コードと呼ばれる暗号に変換されたものからできています。そして，再びこれを解析しながら，BASICのプログラムが実行されます。

中間コードに変換されたBASICのテキストを，具体的に**図2**に示しました。ここで，最初の2バイトは，次の行の中間コードがしまわれている先頭アドレスで，リンクポインタと呼ばれます，テキストの終了のときは，ここが00，00になります。次の2バイトは，行番号が16進になったものです。＄B6は，「LINE」の中間コードです。＄28は左かっこ「(」の中間コードで，これはASCIIコードと同じです。次の「＄FE，＄01，＄02」は，テキスト中の数値定数を表しています。＄FEは，定数を示す中間コードです（グラフィック文字の＄FE(▩)が，通常使用できないのは，実はここに原因があります)。＄01は，数値のバイト長で，＄01，＄02は整数，＄04は実数＄08は倍精度を表します。ちなみに＄03は文字，＄F2は行番号です。＄02は，数値で，ここでは整数なので値は「2」になります。＄2Cは「，」の中間コードで，＄29は右かっこ「)」，＄DAは「−」，＄A6は「PSET」の中間コードです。最後の＄00は，この行の終わりを表しています。

この形式どおりに，テキストを書き込んで，テキストの先頭番地，変数テーブル，フリーエリアのアドレスを書き込めば，BASICテキストのできあがりになります。

インタプリタは，＄47に行番号，＄AFにスタックポインタの値を書き込みながら，BREAKチェック，割り込み処理，テキスト終了チェック，トレース処理をしながら，＄D9を読み込みポインタにして，各命令の処理系への分岐を行います。

プログラムのテクニックについて

F-BASICのルーチンには，テキスト作成部，解読実行部（インタプリタ），各処理系の3本柱があります。このうちの解読実行部は，1つの命令実行が終了した後で，各処理系から直接テキスト作成部に飛ばない限り，＄263のサブルーチンに，一度飛びます。しかし，これは単にRTS命令のみのサブルーチンで，実際にはなにもしません。なぜこんなことをしているのかというと，ここを書き換えることで，BASIC解読実行部の機能拡張ができるようにするためなんです。そこで，せっかくこういうものがあるのだから，使わにゃソン！　というわけで，＄263から，こちらの作ったサブルーチンにJMPしてくるように書き換えます。これでBASICの拡張ができます。

さて，次にBASICプログラムの実行を，どうやって機械語から呼び出しているのかというと，実は＄D9にテキストの先頭番地−1の値を書き込み，すべてのファイルを閉じて，変数をすべて初期化してから，BASICの解読実行部にJMPするだけでいいのです（スタックは，念のために通常レベルに戻しています)。

BASICの実行中や，ダイレクトコマンドの実行後も，拡張ルーチンをコールしていますが，暗黙ENDかどうかチェックして，そうでないときは，そのまま復帰します。

それから，BASICのテキストを，機械語で作成する方法ですが，これは入力された情報（グラフィックカーソルの位置など）によって，もとのテキストの一番終わりに，F-BASICのテキストと同じ形式のものを書き込んでいるだけのことです。ただし，

図1　F-BASIC構成図

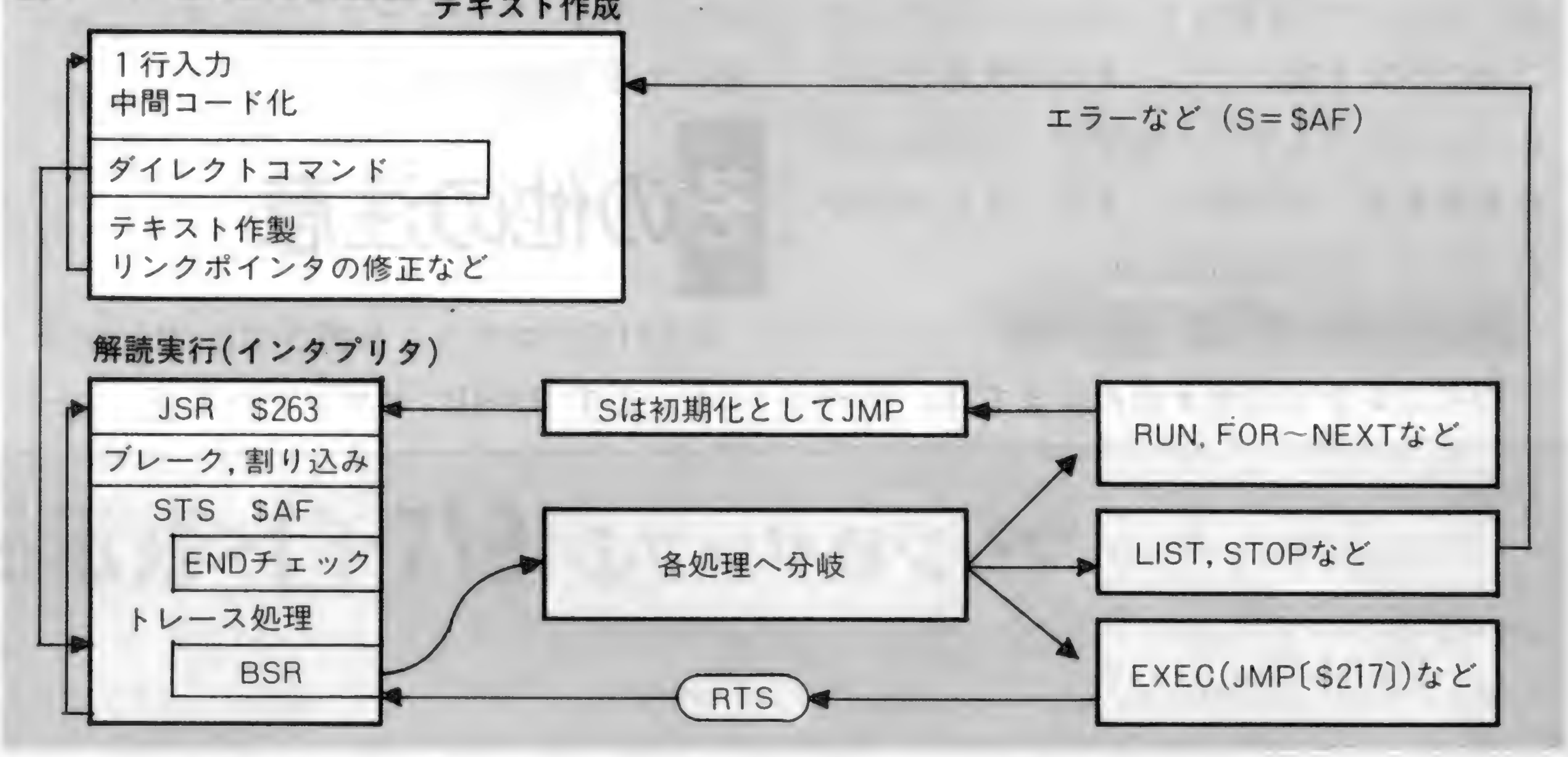


図2　中間コードの例

```
$0CA4   00,0C,C5,00,14,B6,28,FE,01,02,2C,FE,01,00,29,DA
        リンク  行番号 LINE (    2     ,    0     )
        ポインタ

$0CB4   28,FE,02,02,7F,2C,FE,01,C7,29,2C,A6,2C,FE,01,07
        (      639      ,   199    ) , PSET ,    7

$0CC4   00,0C,CE……
        1行の終わり
```

この位置には，変数テーブルがありますが，これ以降の実行には必要ないので，このプログラムでは，変数テーブルは消去してあります。

そして，新たに書き込んだテキストの実行は，なんとBASICのインタプリタをそのまま使っています。つまり，テキストを新たに書き込んで そのままRTSすると，インタプリタは，新しいテキストを，前のテキストに続くものとして そのまま実行を続けてくれるというしかけです。

拡張ルーチンを終了させるときは，$263をもとどおりに書き直して，そのままインタプリタにRTSします。テキストが終了しているため，インタプリタは実行を終了して，テキスト作成部へ戻ってきます。

ところで，BREAKキーでもテキスト作成部へ戻ってきますが，これはキー入力のときに，BREAKキーチェックサブルーチンをコールしているためです。拡張ルーチン中でも，BASIC実行中にBREAKしたのと同じことになります（ただし拡張されたままですが）。

最後に入出力の話をしておきます。

メニュ表示や，REM文の入力データ表示などは，サブコマンドのSYMBOLのXORを使って表示しています。サークルの大きさを設定するときに表示される点線状の円は，内部ルーチンを呼んで，XORで表示しています。これは，カラー，機能を$FB，$C2に設定して，$F3，$F4に半径，$F5，$F6に比率を設定し，$FDを$01にして，Dレジスタに開始位置，$FE，$FFに終了位置を，$180に$FFまたは$00を書き込んで，$E5F8（FM-8は$E75C）をコールします。

キー入力は，すべてBIOSのKEYINを用いています。座標入力のほうは，サブコマンドのグラフィックカーソルによる入力を，BIOSのSUBINを使って実行しています。入力のときは，$100～を入力用データバッファとして使用しています（**図3**）。この部分は，F-BASICでも，入出力バッファに使用しているので，安心して使えます。

終わりに

途中で気がついたのですが，FM-8では，KEY LISTをとると，キャラクタカーソルが表示されます。これはPRINT命令では消えないので，カーソルを表示したままにしたいときには役に立ちます。

このプログラムは，最初のうちは，インタプリタを写しとって，実行させていたのですが，FOR～NEXT，WHILE～WENDなどでは，勝手にインタプリタにJMPしてしまうので，使いものにならなくなってしまい，しかたなく拡張フックを使用することとなりました。でも，結果的にはこの方がよいプログラムになったと思います。

BASICジェネレータ付きのグラフィックツールは，いろいろ出回っていますが，リアルタイムのジェネレータで，ここまでコンパクトなものはないと自負しています。特に，REM文まで生成してくれるジェネレータというものは，後のデバッグのときに，たいへん便利です。

なお，サブコマンドを**図4**に，FM-7,8の変更点を**表1**に示しておきます。

それでは，このプログラムを，グラフィックを作る際のよきパートナーとしてもらえることを願っています。

○参考文献
『F-BASIC解析マニュアルフェーズII』（秀和システムトレーディング株式会社）
『FM-7ユーザーズマニュアルシステム仕様』（富士通）

図3 BIOS

RCB KEYIN

0	21
1	エラーステータス
2,3	データバッファ先頭アドレス
4,5	2

RCB SUBIN

0	17
1	エラーステータス
2,3	データバッファ先頭アドレス
4,5	データバイト数
6,7	入力バイト数

RCB SUBOUT

0	16
1	エラーステータス
2,3	データバッファ先頭アドレス
4,5	データバイト数

図4 サブコマンド

SYMBOL

0,1	——
2	$19
3	カラー
4	機 能
5	角 度
6	横倍率
7	縦倍率
8,9	X座標
10,11	Y座標
12	文字数
13～	文字列

GRAPHIC CURSOR

0,1	——
2	$1F
3	カーソルの色
4	要求する座標数
5,6	初期座標X
7,8	〃 Y

GRAPHIC CURSORの復帰情報

0	エラーコード
1,2	——
3,4	X1
5,6	Y1
7,8	X2
9,10	Y2
⋮	⋮

表1 変更点

アドレス	FM-7	FM-8
$603A	CE81	D145
$603D	8F51	9414
$6046	C63D	C985
$616B	92A0	9740
$64A4	92	E3
$64A6	93	E1
$64AC	E2	91
$64AE	90	8F
$65CA	DB59	DD58
$661F	DB59	DD58
$66BA	E5F8	E75C

リスト●グラフィックプログラムジェネレータ

©1984 A.FURUTA

ADDR	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	=S
6000	6F	8D	05	E5	8E	01	40	9F	EF	8E	00	64	9F	F1	8E	00	53
6010	32	9F	F3	8E	00	73	9F	F5	8E	02	7F	9F	F7	8E	00	C7	53
6020	9F	F9	86	7E	B7	02	63	30	8D	00	1D	BF	02	64	9E	47	9C
6030	30	01	27	01	39	9F	47	32	62	BD	CE	81	BD	8F	51	9E	53
6040	33	30	1F	9F	D9	7E	C6	3D	9E	47	8C	FF	FF	26	01	39	4A
6050	10	9E	D9	A6	A0	27	01	39	EC	A4	27	01	39	30	A8	46	3D
6060	BF	03	3C	11	BC	03	3C	10	25	00	FA	10	BF	03	3C	9E	E5
6070	47	30	0A	BF	03	3E	8C	FA	00	10	24	00	E8	10	8E	03	C4
6080	40	A6	8D	05	64	84	DF	81	4C	27	04	81	44	26	25	17	5E
6090	05	59	AE	C4	9C	EF	26	06	AE	42	9C	F1	27	16	A6	8D	74
60A0	05	47	84	DF	81	4C	26	06	17	00	E9	16	00	83	17	01	59
60B0	9E	16	00	7D	30	8D	04	44	17	04	17	17	03	F6	A6	8D	AB
60C0	05	27	81	0C	27	35	81	30	25	EA	81	38	24	08	80	30	6A
60D0	A7	8D	05	74	20	DE	84	DF	81	45	26	03	16	00	86	81	1A
60E0	46	26	18	30	8D	04	66	17	03	E8	17	03	C7	81	36	24	69
60F0	F9	80	30	25	F5	A7	8D	05	5D	20	B9	C6	09	30	8D	00	BE
SUM	8C	DD	70	01	30	05	3F	72	49	EC	52	F6	AC	49	41	D3	46

ADDR	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	=S
6100	15	A1	80	27	07	30	02	5A	26	F7	20	A8	EC	84	30	8D	02
6110	00	02	AD	8B	20	1B	0C	00	5D	4C	00	68	52	00	9A	50	CE
6120	00	DC	42	00	86	44	01	38	53	01	75	43	02	5A	41	02	CC
6130	5A	BE	03	3E	9F	A6	6F	A0	6F	A0	6F	A0	1F	20	83	03	90
6140	3C	1F	01	10	BE	03	3C	CE	03	3C	F3	03	3C	83	00	02	2D
6150	FD	03	3C	A6	C0	A7	A0	30	1F	26	F8	10	9F	35	10	9F	E9
6160	3B	10	9F	3D	39	86	39	B7	02	63	8E	92	A0	BF	02	64	20
6170	39	86	A4	A7	A0	CC	00	00	17	04	E2	39	17	04	70	86	BD
6180	B6	A7	A0	17	04	A4	86	DA	A7	A0	17	04	9D	17	04	B9	EF
6190	17	04	B0	39	86	B6	A7	A0	20	EC	8D	E0	8E	2C	42	AF	AB
61A0	A1	A6	8D	04	44	85	20	26	04	86	46	A7	A0	39	8E	3A	FF
61B0	8D	AF	A1	30	8D	03	BE	17	03	18	17	02	F7	81	08	26	4C
61C0	0E	B6	01	0C	81	10	27	F2	7A	01	0C	33	5F	20	EB	81	20
61D0	20	25	0B	A7	C0	7C	01	0C	11	83	01	34	25	DC	CE	01	D9
61E0	1D	B6	01	0C	80	0F	4A	26	01	39	E6	C0	E7	A0	20	F6	5C
61F0	17	03	F8	86	B3	A7	A0	17	04	30	30	8D	03	88	17	02	3E
SUM	79	89	75	53	72	55	B0	D9	DE	C4	83	12	21	9A	DC	AF	97

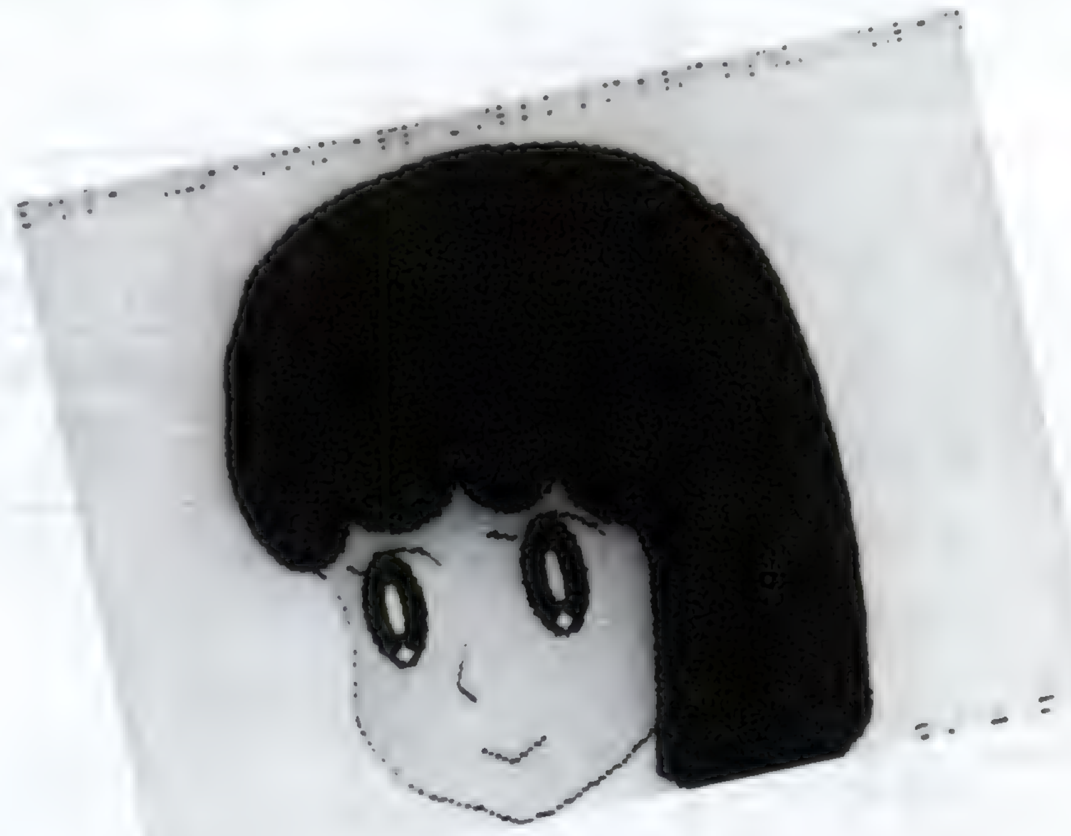
ADDR	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	=S
6200	D1	17	02	B0	81	08	26	10	B6	01	0C	81	10	27	F2	80	46
6210	02	B7	01	0C	33	5E	20	E9	81	30	25	16	81	38	24	12	3B
6220	1F	89	86	2C	ED	C1	7C	01	0C	7C	01	0C	11	83	01	2F	DE
6230	25	CF	CE	01	1D	B6	01	0C	80	0E	80	02	26	01	39	E6	F9
6240	C0	E7	A0	E6	C0	C0	30	17	04	1E	20	EE	17	03	9C	A6	80
6250	8D	04	03	81	01	27	2A	4D	26	13	8D	11	86	DE	A7	3E	D4
6260	86	3A	A7	A0	33	5C	8D	05	86	DF	A7	3E	39	86	A6	A7	7E
6270	A0	17	03	B6	31	3F	17	03	CA	17	03	CD	86	29	A7	A0	A1
6280	39	86	A7	A7	A0	17	03	A2	39	17	03	5F	AE	C1	9F	EF	18
6290	AE	C1	9F	F1	30	8D	02	FF	17	02	37	17	02	16	81	30	ED
62A0	25	F9	81	34	24	F5	80	30	34	02	30	8D	03	0A	17	FF	B2
62B0	06	B6	01	0C	81	10	27	F2	CE	01	00	9E	EF	AF	48	9E	64
62C0	F1	AF	4A	35	02	A7	45	A6	4C	80	10	A7	4C	30	C8	1D	97
62D0	E6	80	E7	88	EF	4A	26	F8	17	01	D9	81	0D	27	46	4F	67
62E0	F6	01	06	17	01	5D	10	83	00	01	2B	03	F7	01	06	4F	81
62F0	F6	01	07	17	01	53	10	83	00	01	2B	03	F7	01	07	FC	26
SUM	5F	89	AA	69	4B	A9	F8	D9	F2	81	B2	7E	0D	5C	7A	45	8B

ADDR	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	=S
6300	01	08	17	01	4A	FD	01	08	FC	01	0A	17	01	57	FD	01	E5
6310	0A	A6	8D	02	D4	81	30	26	0A	86	01	BB	01	05	84	03	C3
6320	B7	01	05	20	B3	10	8E	03	40	86	C5	A7	A0	CE	01	08	DA
6330	17	02	F7	8E	2C	22	AF	A1	B6	01	0C	8E	01	0D	E6	80	01
6340	E7	A0	4A	26	F9	86	22	A7	A0	F6	01	06	17	03	21	F6	0D
6350	01	07	17	03	1B	17	02	EB	F6	01	05	17	03	12	17	02	82
6360	E8	E6	8D	02	F1	C1	05	26	04	C6	D7	E7	3F	39	A6	8D	6D
6370	02	77	B7	01	80	86	C3	A7	A0	17	02	6F	AE	C1	9F	EF	C6
6380	AE	C1	9F	F1	0F	FE	0F	FF	17	03	08	17	02	3B	17	03	AA
6390	02	A6	8D	02	54	81	0D	27	1E	DC	F3	17	00	A5	DD	F3	B9
63A0	DC	F5	17	00	A4	DD	F5	DC	EF	17	00	A3	DD	EF	DC	F1	7C
63B0	17	00	B2	DD	F1	20	D1	B6	01	80	84	02	26	18	AE	8D	BE
63C0	00	5D	9F	FE	CE	00	FE	17	00	57	CE	00	FF	17	00	51	69
63D0	9E	FE	AF	8D	00	49	CE	00	EF	17	02	4E	86	2C	A7	A0	3E
63E0	DC	F3	17	02	78	17	02	5B	DC	F5	17	02	89	B6	01	80	7E
63F0	84	02	27	07	8E	2C	2C	AF	A1	20	10	4F	D6	FE	58	58	ED
SUM	4C	61	C6	41	4E	9C	36	0A	C7	DB	31	EC	93	24	63	3D	F4

ADDR	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	=S
6400	17	02	73	4F	D6	FF	58	58	17	02	6B	8E	2C	4E	AF	A1	3C
6410	B6	01	80	84	20	26	04	86	46	A7	3F	17	02	2B	39	00	34
6420	00	17	02	6F	17	01	A2	17	02	69	A6	8D	01	BB	81	0D	41
6430	27	10	4F	E6	C4	34	40	17	00	09	35	40	C4	3F	E7	C4	E7
6440	20	DF	39	33	8D	00	4D	20	32	33	8D	00	4F	20	2C	33	25
6450	8D	00	51	8D	26	4D	2B	0A	10	93	F7	22	01	39	CC	00	D5
6460	00	39	DC	F7	39	33	8D	00	43	8D	10	4D	2B	0A	10	93	0A
6470	F9	22	01	39	CC	00	00	39	DC	F9	39	1F	01	C6	04	A6	F8
6480	8D	01	66	A1	C1	27	06	5A	26	F9	1F	10	39	E6	5F	30	D9
6490	85	1F	10	39	06	0A	02	F6	1C	01	1D	FF	19	F6	1A	0A	61
64A0	1E	FF	1F	01	92	0A	93	F6	36	01	34	FF	E2	F6	90	0A	3E
64B0	38	FF	32	01	30	8D	00	14	AD	9F	FB	FA	17	01	0A	30	CE
64C0	8D	00	09	AD	9F	FB	FA	A6	8D	01	1E	39	10	00	01	00	73
64D0	00	5D	34	10	CE	01	00	30	8C	16	86	0C	E6	80	E7	C0	E1
64E0	4A	26	F9	35	10	A6	84	4C	E6	80	E7	C0	4A	26	F9	39	D3
64F0	00	00	19	07	04	00	01	01	00	00	00	00	50	20	45	29	04
SUM	D9	05	C1	ED	93	44	5D	EC	E4	98	48	0D	4A	35	95	74	05

ADDR	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	=S
6500	6E	64	20	30	29	2D	37	29	3A	43	4F	4C	4F	52	20	46	F7
6510	29	75	6E	63	74	69	6F	6E	20	52	29	65	6D	61	72	6B	D4
6520	20	4C	29	69	6E	65	20	42	29	6F	78	20	44	29	6F	74	B3
6530	20	50	29	61	69	6E	74	20	53	29	79	6D	62	6F	6C	20	24
6540	43	29	69	72	63	6C	65	20	41	29	72	63	20	27	30	3A	8B
6550	50	53	45	54	2C	31	3A	50	52	45	53	45	54	2C	32	3A	3E
6560	4F	52	2C	33	3A	41	4E	44	2C	34	3A	58	4F	52	2C	35	01
6570	3A	4E	4F	54	20	10	52	45	4D	2E	3F	28	3C	9B	3A	45	2A
6580	4E	44	29	20	20	20	10	43	4F	4C	2E	3F	28	3C	9B	3A	AF
6590	45	4E	44	29	20	20	20	20	52	41	44	2E	3F	20	28	20	2C
65A0	30	2D	33	20	29	20	20	20	30	3A	1C	20	31	3A	1E	20	88
65B0	32	3A	1D	20	33	3A	1F	20	10	53	54	52	2E	3F	28	3C	2F
65C0	9B	3A	45	4E	44	29	20	20	20	BD	DB	59	8D	04	5D	27	3B
65D0	F8	39	30	8C	14	AF	8C	0D	30	8C	08	AD	9F	FB	FA	EC	3A
65E0	8C	07	39	15	00	65	E9	00	02	00	00	C6	01	20	02	C6	E0
65F0	02	CE	01	00	8E	00	00	AF	C1	86	1F	A7	C0	A6	8D	00	0E
SUM	09	D2	75	22	DF	2E	7D	71	D6	E6	8B	B8	14	25	24	C2	8B

ADDR	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	=S
6600	47	ED	C1	9E	EF	AF	C1	9E	F1	AF	C1	30	8C	14	AD	9F	0D
6610	FB	FA	CE	01	03	FC	03	12	26	01	39	6F	8C	CB	BD	DB	96
6620	59	39	11	00	01	00	00	0B	00	0B	86	28	A7	A0	EC	C1	5C
6630	DD	EF	8D	29	86	2C	A7	A0	EC	C1	DD	F1	8D	1F	86	29	51
6640	A7	A0	39	E6	8C	02	20	28	07	30	8C	0B	A6	8C	07	E6	29
6650	86	86	2C	ED	A1	39	00	A6	A7	DF	DE	E0	E0	4D	27	08	45
6660	8E	FE	02	AF	A1	ED	A1	39	8E	FE	01	AF	A1	E7	A0	39	42
6670	86	2C	A7	A0	20	F2	10	83	00	00	27	F4	8E	2C	FE	AF	20
6680	A1	8E	04	89	AF	A1	6A	3F	58	49	24	FA	44	56	ED	A1	9C
6690	6F	A0	39	F6	01	80	9E	FE	34	74	86	07	97	FB	86	04	AC
66A0	97	C2	7F	01	80	C6	01	D7	FD	8E	00	FE	8D	16	34	06	5D
66B0	8E	00	FF	8D	0F	DD	FE	35	06	BD	E5	F8	35	74	F7	01	7A
66C0	80	9F	FE	39	A6	84	44	44	44	E6	84	C4	07	39	00	00	BA
66D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
66E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
66F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
SUM	6E	EE	F4	30	4C	39	87	72	12	77	02	01	A5	9E	46	E6	F9



パソコンの計算精度について

```
10 A!=1/3
20 B!=.333333
30 PRINT CDBL(A!),CDBL(B!)
```

プログラムは完璧なのに，答えの数値がなぜか違ってしまうといった経験をお持ちの方も多いことでしょう。本稿は，パソコンの計算は，どの程度まで正確か？　誤差の出る原因と対策について，お話ししましょう。

阿部　正

まずは，パソコンの計算精度とはどういうものか？　について，ふれておきましょう。

次のプログラムを実行してみてください。結果はどうなるでしょうか？

```
10 A=1/3
20 A=3*A
30 PRINT A
```

結果は「1」になりますね。当たり前だと思われるかもしれませんが，これが当たり前ではないのです。

パソコンというのは，分数の計算がまったくできないのです。

このため，パソコンに「1／3」の計算をさせると，ある桁数だけ計算して，そこで計算を打ち切ってしまいます。このときのAの値は「0.3333…3」になります。

次にAに3をかけると，0.9999…9になるのですが，ここである桁数で四捨五入が行われて，Aの値が「1」になります。このときの『ある桁数』を『**有効桁**』といいます。この有効桁が大きいほど，ほとんど等しい値を区別する能力，すなわち精度が高くなります。

『有効桁』は，単精度の場合は6桁で，倍精度の場合は16桁です。これを『**有効数字6桁**』，および『**有効数字16桁**』，または『**6桁の精度**』，『**16桁の精度をもつ**』，といいます。

では，この有効数字がなぜ6桁や16桁なのかを，パソコンの内部をのぞいて，調べてみましょう。

パソコンの内部では，2進数しか扱っていません。そのため，実数もすべて，パソコン内部では2進形式でしまわれています。VARPTR という関数を用いると，このしまわれている場所がわかるので，次のプログラムでのぞいてみましょう。

```
10 A!=10
20 ADA=VARPTR(A!)
30 FOR I=0 TO 3
40 PRINT HEX$(PEEK(ADA+I))
50 NEXT
```

結果は「84, 20, 0, 0」になります。なぜこんな値になるかを，手短かに説明しましょう。

$10=8+2$

$=2^3+2^1$

$=1\times2^3+0\times2^2+1\times2^1+0\times2^0$

$=2^4\times(1\times2^{-1}+0\times2^{-2}+1\times2^{-3})$

となるので，

指数部$=4_{(10)}=100_{(2)}$

仮数部$=1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

$=0.101_{(2)}$

となります。ここで，$_{(10)}$，$_{(2)}$ はそれぞれ10進数表示，2進数表示を示します。

指数部が頭の8ビット，仮数部がその後ろに続きます。これをそのまま書くと，以下のようになります。

指数部			仮数部					
0000	0100	.	1010	0000	0000	0000	0000	0000

↑仮想小数点

これに，ゲタはかせと全体の符号付けが加わって，次のような形になります。

ゲタ（正のとき1，負のとき0）

1000	0100		0010	0000	0000	0000	0000	0000

符号（正のとき0，負のとき1）

どうですか，16進形式に直すと「84, 20, 00, 00」になりましたね。ゲタと符号付けについては，話すと長くなるので，こういうものかと思っておいてください。

ついでに「10　A！＝−0.3」と書き換えてRUNしてください。結果は，「7F, 99, 99, 9A」となります。この場合は，

$-0.3=-10^{-1}\times(0.10011001\cdots)_{(2)}$

となるので，まず，次の形になります。

11111111	.	10011001	10011001	10011001	1001……

これに，ゲタと符号付けと四捨五入（2進では0捨1入）が加わって，次の形になります。

ゲタ（負なので0）　繰り上がり

01111111	.	10011001	10011001	10011010

符号（負なので1）

これを16進形式で書くと，「7F, 99, 99, 9A」になります。

単精度ではこのように，指数部は8ビットの精度があります。

$2^7=128$で,$2^{\pm128}\fallingdotseq3.4\times10^{\pm38}$となるので，10進形式では$10^{\pm38}$の範囲の数をあつかえます。$10^{38}$を超えるとオーバーフローになり、$10^{-38}$より小さいときは0とみなされます。

仮数部のほうは，24ビット（3バイト）の精度があるので，$2^{24}\fallingdotseq1.7\times10^7$より，10進形式では7桁の精度を持ちます。つまり，機械内部では，7桁の精度をもちますが，表示は6桁で表示されます。

倍精度の場合は，全体が8バイトで，指数部1バイト，仮数部7バイトです。仮数部が拡張されて7バイト＝56ビットになったため，$2^{56}\fallingdotseq7.2\times10^{16}$で，10進形式では仮数部が16桁の精度を持ちます。

今までのことを簡単にまとめてみましょう。

パソコンでは，実数を扱うときには，分数を使わずに，すべて小数で処理している。しかし，無限の小数を扱うことはできないので，4バイト（単精度）もしくは8バイト（倍精度）の長さで表せる範囲に限られている。このため，似たような値を区別できる能力には限りがあり，この能力の尺度を精度という。

*　　　*　　　*

それでは，この精度が実際の計算にどうかかわってくるのでしょうか。パソコンでは，精度の範囲での計算しかできません。このため実際の値とパソコン内部の値とが，違ってくる場合があります。

この値の差のことを**誤差**といいます。一般に精度が高いほど誤差は少ないのですが，計算の手順によっては，誤差がとんでもなく大きなものになるときがあります。

これから先は，誤差の種類を，その実例を挙げて紹介していきましょう。そして，誤差を少なくおさえるにはどうしたらよいのかをお話ししましょう。

I 誤差のいろいろ

(1) 10進2進変換誤差

◇プログラム①

```
10 A!=.1
20 A#=CDBL(A!)
30 PRINT A#
```

◇結果

```
.1000000014901161
```

パソコンでは，内部にしまわれる値は2進数に変換されます。この変換のときに生じる誤差を『**10進2進変換誤差**』といいます。

0.1という数は，10進数では精度の範囲内のようにみえますが，これを2進数で表すと，$0.1_{(10)}=0.000110011001$……という無限に続く小数になってしまうので，途中で四捨五入されて誤差が生じます。このときの誤差は，仮数部で$2^{\pm24}$以内なので，これだけでは2進→10進への変換の際の四捨五入で消えてしまい，ほとんど問題になりません。

しかし，わずかな誤差でも上のプログラムのように精度を高めると，その違いがはっきりでてきますので，たとえ見た目でどんなに精度が高そうに見えても，単精度にはそれなりの精度しかないことに注意する必要があります。

下のプログラムでの誤差も，この10進2進変換誤差によるものです。

◇プログラム②

```
10 A!=1E+38
20 PRINT CDBL(A!)
```

◇結果

```
1.000000272264713D+38
```

◇プログラム③

```
10 A!=.1
20 B!=0
30 FOR I=1 TO 500
40 B!=B!+A!
50 NEXT
60 PRINT B!
```

◇結果

```
49.9998
```

(2) 丸め誤差

◇プログラム④

```
10 A!=.3333333:PRINT A!
```

◇リストをとると,

```
10 A!=.333333:PRINT A!
```

◇結果

```
.333333
```

このように，小数を途中で四捨五入したときに生じる誤差のことを『**丸め誤差**』といいます。広い意味では，前の10進2進変換誤差や，計算中の丸めによる誤差もこのうちに入りますが，ここでは，数値の入出力の際に値を丸める（途中で四捨五入する）ときに生じる誤差のこととします。

上の**プログラム④**のように，打ち込んでリストをとってみると，値が丸められているのがわかります。これは，F-BASICで単精度定数は7桁までの精度で格納され，6桁までで表示されると決まっているからです。ちなみに，プログラム中の定数は，パソコン内部では下のように格納されています。

「FE,　04,　80, 4C, ……」
↑実数のしるし　↑バイト数　↑2進変換された定数値

このため，リストをとっただけでは本当の.333333なのか，.3333333が丸められているのかわからない場合もでてきます。これについては，後の(8)でふれます。

とにかく，仮数部の精度を超える長さの数値が入出力されるときには，値が丸められて誤差が生じるので注意してください。

次のプログラムの結果も，この丸め誤差が原因となるものです。

◇プログラム⑤

```
10 A!=.1
20 B#=CDBL(A!)
30 C#=VAL(STR$(A!)+"#")
40 PRINT B#,C#
```

◇結果

```
.1000000014901161 .1
```

(3) 加減乗除の誤差

◇プログラム⑥

```
10 PRINT 1+1E-8,1-1E-8
20 PRINT .999999*3,1/3
```

◇結果

```
1                 1
3                 .333333
```

加減乗除を行うときに，結果が仮数部の精度を超えていたら，値が丸められて誤差が生じます。

かけ算やわり算に関しては，表示がもともと四捨五入されているので，誤差をためこまない限り，精度の低下はそれほど心配することはありません。しかし，たし算やひき算では，$(A+B)-A \fallingdotseq B$，$(A-B)+B \fallingdotseq A$などという場合もおこってくるので，絶対値の違いすぎる2数の加減は，できるだけやらない方がいいでしょう。

一般に，演算の回数が増えれば増えるほど，誤差は大きくなると思っておいてまちがいありません。

(4) 関数の誤差

◇プログラム⑦

```
10 PRINT SQR(2),SQR(2)^2
```

◇結果

```
1.41421          2
```

$\sqrt{2}$の実際の値は1.41421356…と無限に続く小数ですが，FM-7，8で与えられる値は単精度で与えられます。SQRに限らず，EXP，SINなど，すべての数値関数の精度は単精度しかありません。このため，FM-7/8で関数や単精度を用いる場合は，倍精度で計算してもほとんど無意味です。また，FOR～NEXTループにも倍精度変数は使えません。

FM-11では，SQR(2#)で，倍精度の答えがでます。

(5) 桁おち

◇プログラム⑧

```
10 A!=1.00001
20 B!=1
30 PRINT A-B
```

◇結果

```
9.89437E-06
```

誤差のうちでいちばんやっかいなものが，この『**桁おち**』です。ほかのものは，誤差がたまるか，単精度を倍精度に変換しない限り，表示の精度は保たれます。しかし，非常に近い2数を加減したときには『**精度の低下**』と呼ばれるほどの，精度の範囲を超える誤差が生じます。

これは，内部で2進変換して丸められていることによるものですから，パソコンでは，近接した2数の減算は可能な限り避けてください。

とくに，$|A| \gg |B|, |C|$ のとき $(A+B)-(A+C)$ の形になるプログラムに対しては，$(A-A)+(B-C)$の形になるように変形してください。

(6) 打ち切り誤差

◇プログラム⑨

```
10 X=1:ATAN=0:CO=1
20 FOR I=1 TO 10001 STEP2
30 ATAN=ATAN+CO*(X^I)/I
40 CO=-CO
50 NEXT
60 PRINT ATN(X),ATAN
```

◇結果

```
.785398          .78545
```

E＊P(X)，SIN(X)，ATN(X)…etc は無限級数で表示できます。

たとえば，ATN(X)に $x-\frac{x}{3}^3+\frac{x}{5}^5+\cdots$

を求めるときには，有限回で打ち切って，それを近似値とします。このときに生じる誤差を『**打ち切り誤差**』といいます。

級数の収束が遅いと，打ち切り誤差が精度を超えることもあります。ここではわざと収束の悪い例を使っていますが，収束が遅い級数はなるべく使わない方がいいでしょう。余談ですが，ATN(1)$=\frac{\pi}{4}$ですから，円周率πは4＊ATN(1)で近似できます。

(7) 単精度,倍精度,整数の変換誤差

◇プログラム⑩

```
10 A#=.49999999#
20 A!=A#:A%=A#:A#=A!
30 PRINT A%,A!,A#
```

◇結果

```
 1              .5              .5
```

倍精度→単精度，および単・倍精度→整数のときには，変換後の精度の範囲で四捨五入されます。また，単精度→倍精度の変換では，そのまま拡張されますから，10進→2進のときの丸め誤差が，倍精度上で目に見える誤差になってでてきます。このような誤差を『**変換誤差**』といいます。

単精度を倍精度に変換しても,それ以降の計算の精度が高くなるだけでそれまでの精度は保たれますから，解の精度は単精度です。

整数→単・倍精度の変換は，誤差は生じませんから精度も高くなります。もっともこれは例外的な場合ですから，整数は整数のまま,単精度も同様,ほかへの変換はやめておいた方がいいでしょう。

(8) 見かけ上の誤差

◇プログラム⑪

```
10 A!=CSNG(.5555545#)
20 B!=CSNG(.5555554#)
30 PRINT A,B
```

◇結果

```
 .555555        .555555
```

単精度がパソコン内部で7桁の精度でしまわれ，6桁で表示されることは前述しました。このため，上のように画面に表示しただけでは同じ値なのか，違うかは区別できません。極端な話をすれば，表示ではAとBが同じ値なのに，IF A＝B THEN～の処理をしてくれないといった場合も起こります。

下の例も，この**見かけ上の誤差**で起こるものです。

◇プログラム⑫

```
10 A!=CSNG(111111.1#)
20 B!=111111!
30 PRINT A,B,A=B
```

◇結果

```
 111111         111111          0
```

(9) あふれ

◇プログラム⑬

```
10 A!=1.7E+38
20 B!=5.8E-39
30 PRINT B/10,10*A
```

◇結果

```
 0         Overflow In 30
```

精度の範囲を超える値を無理に代入しようとしたときに生じるのが「あふれ」で，値が大きすぎる場合の上位桁あふれでは「Overflow」を表示してエラーになり，値が小さすぎる場合の下位桁あふれでは，値が0になります。後者では「Underflow」のエラーとなる機械もありますが，F-BASICでは10^{-38}以下の数はすべて0とみなします。これは内部の値が「00,00,00,00」のときに値が0と等しくなるようにつくられているためです。ただし，指数部が「00」であれば，すべて「0」とみなすので注意してください。

*　　　*　　　*

以上がおもな誤差です。つまり，パソコンでは，精度を超えた場合には正確な計算は絶対にできません。たとえば，20桁くらいの数にいたっては，入力も表示もできません。

ただし，これは精度を超えた場合で，よほど複雑な科学計算や実務計算でない限り，6桁以上の精度は必要ないでしょう。誤差の累積と，精度の低下に注意してプログラムを作りさえすれば，パソコンでも充分に実用に耐えられるだけの数値計算ができます。

それでは，具体的にプログラム例をつけて，誤差を小さくおさえる方法を示しましょう。

II 桁おち

◇プログラム⑭

```
10 A=1:B=200:C=1
20 X=(B-SQR(B^2-4*A*C))/(2*A)
30 PRINT X
```

◇結果

```
 5.00488E-03
```

◇プログラム⑮

```
10 A=1:B=200:C=1
20 X=(2*C)/(B+SQR(B^2-4*A*C))
30 PRINT X
```

◇結果

```
 5.00013E-03
```

この**プログラム⑭**は，$AX^2+BX+C=0$の解の1つを解くものです。より正確な値は5.000125E-03ですが，**プログラム⑭**では「$B-\sqrt{B^2-4AC}$」のところで桁おち誤差が起こっています。このため**プログラム⑮**のように

$$\frac{b-\sqrt{b^2-4ac}}{2a}\rightarrow\frac{b-\sqrt{b^2-4ac}}{2a}\times\frac{b+\sqrt{b^2-4ac}}{b+\sqrt{b^2-4ac}}\rightarrow\frac{2c}{b+\sqrt{b^2-4ac}}$$

と変形して，桁おちを防ぎます。

一般に，B＞0，|B|≫|A|で，$B-\sqrt{B^2-A}$ の形の計算をすると必ず桁おちが起こりますが，上の方法は桁おちを防げる代表的な例ですから，知っておくと便利でしょう。

III 打ち切り

◇プログラム⑯

```
10 X=1:ATAN#=0:CO=1
20 FOR I=10001 TO 1 STEP-2
30 ATAN#=ATAN#+CO*(X^I)/I
40 CO=CO*(-1)
50 NEXT
60 PRINT ATN(X),CSNG(ATAN#)
```

◇結果

```
.785398        .785448
```

小さい方から加えることと，倍精度に加えることで桁ずれを小さくおさえることによって，打ち切り誤差は少なくなります。**プログラム⑨**の結果と比べてみてください。

IV 誤差の累積と小数

◇プログラム⑰

```
10 M=1000
20 A!=.1:B!=0
30 A!=A!*M
40 FOR I=1 TO 500
50 B!=B!+A!
60 NEXT
70 B!=B!/M
80 PRINT B!
```

◇結果

```
50
```

プログラム③の結果と比べてみると，精度が上がっているのがわかりますね。このように，小数はなるべく使用せずに，適当な数（ここではM）を使って，精度の範囲内での整数で計算すれば誤差は小さくおさえられます。なお，誤差の累積については，おおまかですが単精度で40～400回以内の計算回数なら，表示まで誤差がくいこんでくることはないでしょう。また，誤差をためこんでも，それが次に実行する演算によって生じる誤差よりも小さければ，ためた誤差が消えてしまいます。この辺はプログラマの腕次第です。

V 倍精度の使い方

Iの(4)の関数の誤差のところで，単精度やFM-7/8で関数を用いる場合は，倍精度を使っても無意味だと書きましたが，細かい計算をするときには，倍精度が便利な場合もあります。

(1) 四則のみの場合

◇プログラム⑱

```
10 PRINT CDBL(1/3),1/3#
```

◇結果

```
.3333333432674408  .3333333333333333
```

四則演算のみから成る計算のときは，倍精度で計算すれば，誤差をためこまない限り，16桁の精度で計算できます。

(2) 桁ずれの場合

◇プログラム⑲

```
10 A!=1
20 B!=1E-10
30 PRINT (A+B)-A
```

◇結果

```
0
```

◇プログラム⑳

```
10 A!=1
20 B!=1E-10
30 A#=CDBL(A!)
40 B#=CDBL(B!)
50 PRINT CSNG((A#-B#)-A#)
```

◇結果

```
-1E-10
```

単精度では，それほど大きくない桁ずれならば，倍精度に変換することである程度まで防げます。

プログラム⑲では(A＋B)－A≠Bというおかしな結果が出ていますが，**プログラム⑳**のようにすればご覧のとおりです。

(3) 定数の場合

◇プログラム㉑

```
10 A!=.3333333!
20 B!=.3333333#
```

◇リストをとると，

```
10 A!=.333333
20 B!=.3333333#
```

単精度の変数に6桁を超える数値を代入する場合には，倍精度

のしるし「#」をつけてから代入した方がいいでしょう。このプログラムのままでは，A！に .3333333 の値が入っていますが，一度スクリーンエディットをすると，Aの値は .333333 になってしまいます。「#」をつけておけば，このような心配はありません。

(4) 見かけ誤差を見抜く

◇プログラム㉒

```
10 A!=1/3
20 B!=.333333
30 PRINT CDBL(A!),CDBL(B!)
```

◇結果

```
.3333333432674408  .3333330154418945
```

単精度の見かけ誤差は，内部をのぞかなくても，CDBLで倍精度に変換してみれば見抜けます。単精度の表示は6桁ですが，内部では7桁の精度をもっています。倍精度に変換すると，この7桁の値が画面にでてくるわけです。

(5) 整数のあふれをふせぐ

◇プログラム㉓

```
10 A%=32768!
20 PRINT A%
```

◇結果

```
Overflow In 10
```

◇プログラム㉔

```
10 A#=32768!
20 PRINT A#
```

整数は－32768～＋32767までの精度しかありません。しかし，単精度や倍精度ではそれ以上の精度をもっているので，整数であふれが出る場合でも，倍精度にすれば16桁の精度までは防ぐことができます。

＊　＊　＊

VI 終わりに

以上でパソコンの精度の話はおしまいです。パソコンがいかに不確かなものか，おわかりいただけたと思いますが，同時にまたプログラム次第で正確にもなる，ということもわかっていただけることを祈っております。

Oh!FM 年間定期購読のお知らせ

Oh!FM を毎号確実に入手したいという方に，年間定期購読をおすすめします。近くに書店がない，買いそびれてしまうことが多い，という方に最適です。発売は毎月16日，書店に並ぶよりも早く確実に入手できます。

年間定期講読料　6,000円（送料込）

申込方法　とじこみの郵便振替用紙にてお申し込みください。現金書留など，他の方法では無効となりますので，ご注意ください。

受付月号　毎月10日当社着を締め切りとさせていただきます。したがって，10月号（9月18日発売）より定期購読ご希望の方は，9月10日までにお申し込みください（当社着）。また，各号締切日以前ならば何月号からでも始められますので，必ず振込用紙に○月号からと明記してください。なお，バックナンバーにさかのぼってのお申し込みはご遠慮ください。

いまお申し込みの方にプレゼント！

9月30日（当社着）までにOh!FM年間購読お申し込みの方全員に，「ソフトバンクカード」をプレゼントいたします。本カードは日本全国どの銀行の現金自動払出機もご利用になれません。そのほか，本カードによるいかなる特典もありません。何の役にも立たないカードですが，だいじに保管してください。そして1人でいるときなど，そっと眺めて笑ってやってください。

for FM-7,8,11(BS除く)　6809機械語(マシンランゲージ)マスタ'84

第5回 スタックと特殊なアドレッシング

鶴岡　哲明

この連載もいよいよ第5回，6809機械語のもっともオイシイ所に到達します。まずスタックですが，スタックならたいていのCPUに備わっていますが，6809ほど使いよくありません。また，7章のアドレッシングも6809ならではのものがそろっています。内容的には多少難しいことばかりですが，始めてもう半年，ガッツで取り組みましょう。

第6章 スタック

1 スタックとは

機械語プログラミングにあたり，エレガントなプログラムを書こうとすると，避けてとおれない項目の1つが「スタック」です。

スタックを一言で説明すると，「LIFO(Last In First Out) のバッファ」ということになるのですが，これだけで説明できるなら，こんな大げさなタイトルを付けません。

まずなぜスタックというものが存在するのかについて説明することにします。

プログラムを書いていて，「ちょっとこのレジスタの内容をどこかに保存しておきたいな」と思ったことがきっとあるでしょう。6809は，レジスタが9個(しかもPCやCCなどは勝手に使えない）だけしかないので，各種変数をすべてレジスタに割り当てるわけにはいきません。また，加減算といった演算はアキュムレータを使うので，次の演算を行う前に内容をどこかに残しておく必要が生じます。

こういった場合，後々でも使うような値ならいざ知らず，ちょっととっておきたいだけの値のために特定のアドレスを使うのはバカバカしい話です。このために，データを一時的に記憶するためのしかけがあると助かります（そんなものかな，と思う人もいるかもしれませんが，この章を読み終えた頃には，ナルホド！と思うはずです)。すなわち，**データを一時記憶するために，CPUに備えられたしかけがスタック**なのです。

スタックが一時記憶のために設けられているといっても，データを長時間覚えていられないかというと，そんなことはありません。ただ，スタックは**複数のデータを記憶できますが，出入り口が1つ**なのです。そのため，普通のメモリのようにランダムアクセス（取り出したいデータがいつでも好きなときに取り出せる）が行えないことになります。そのかわり，出入り口が1つなので，「スタックにデータをしまう」という場合，アドレスを細かく指定する必要がありません。そこでもっぱら一時記憶に用いられるというわけです。

ではスタックの動作を見てみましょう。

スタックというのは，データを積み上げるように記憶します(**図1**)。つまり，新しいデータを古いデータの上に積み上げ，データを読み出すときは，積み重なったデータの山の上から取り出すのです（**図2**)。

こうすると，最後に入れた（積んだ）データが最初に出てきます。たとえば，5個のデータを順にスタックに入れると，**図3**のようになるのです。この状態でスタック

図1　スタックの概念

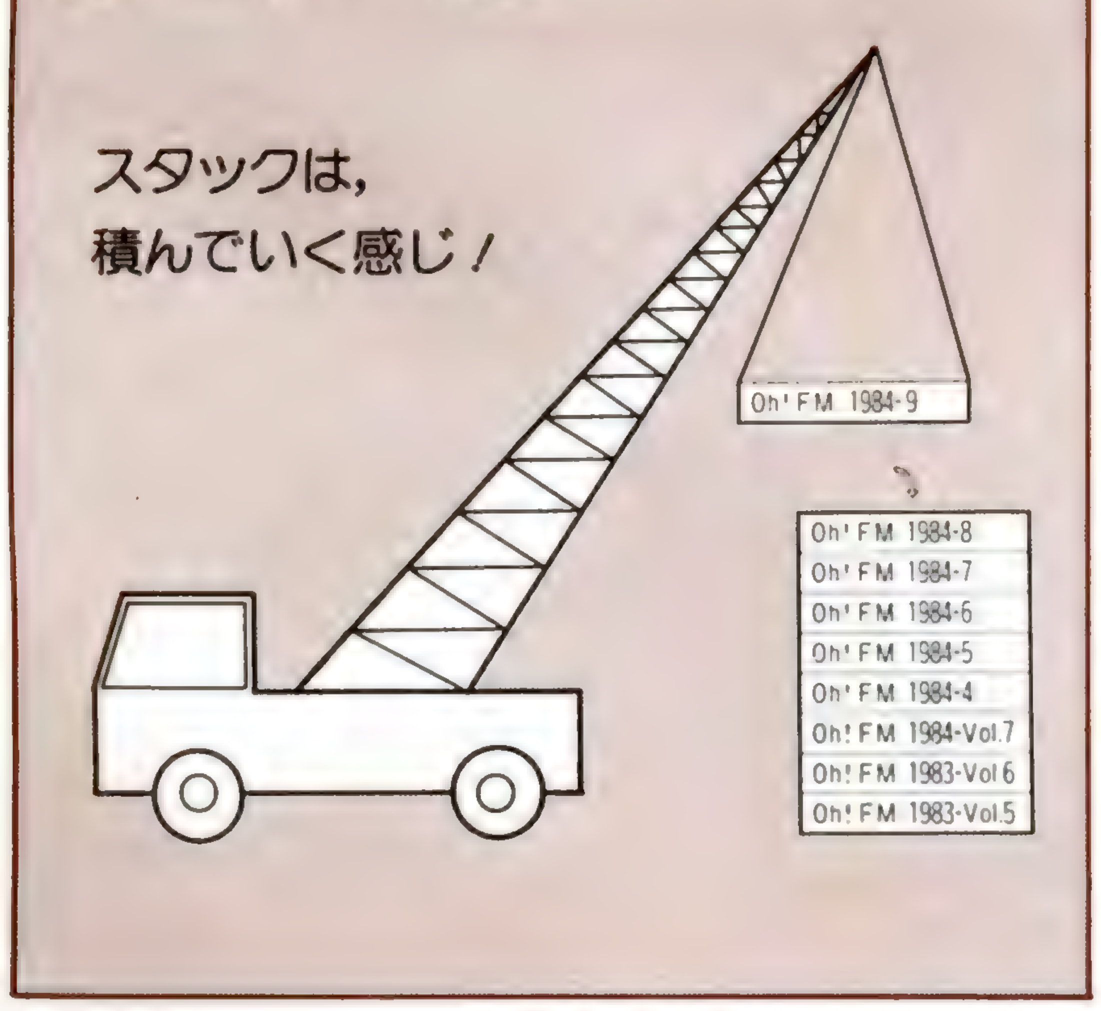


図2　スタックの動作

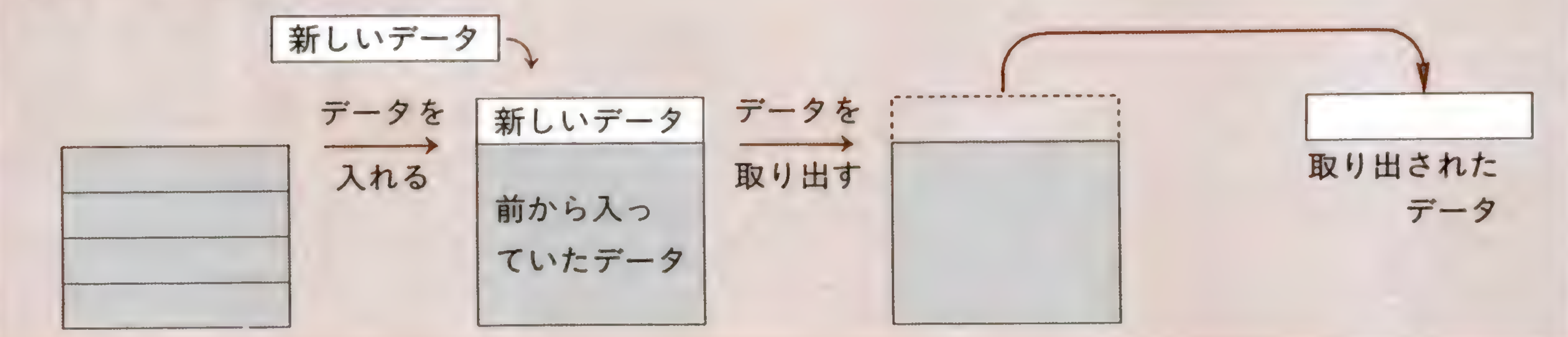
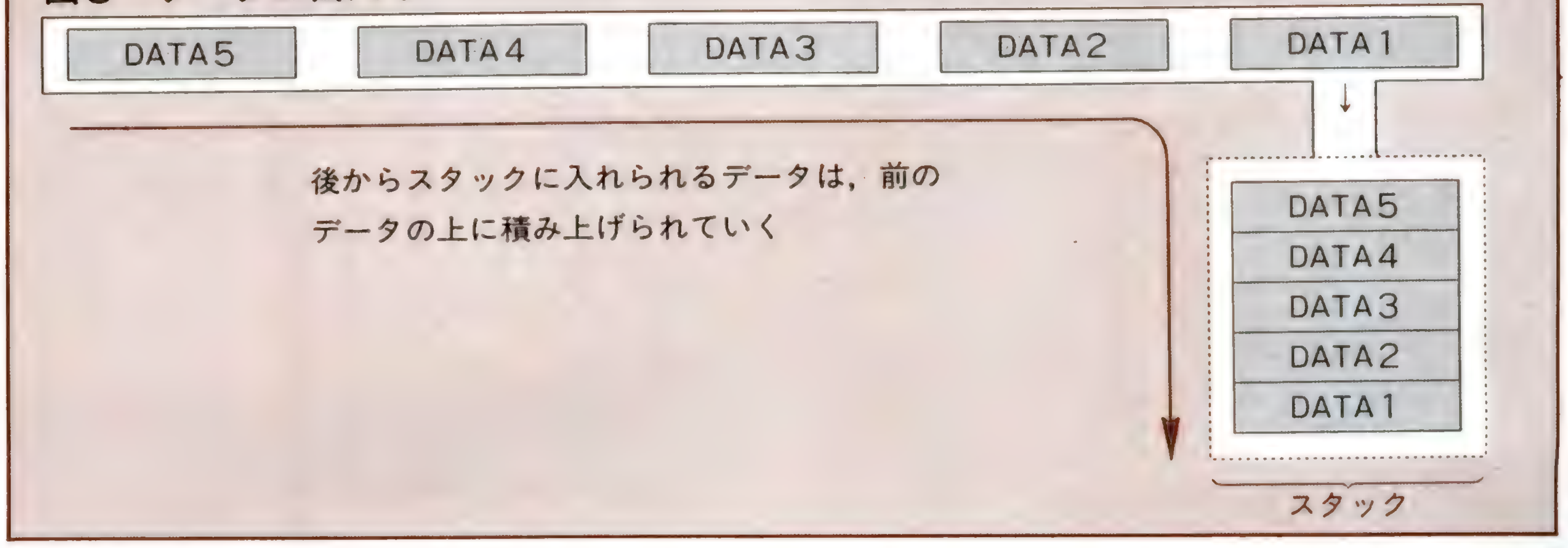


図3　データの出入り



からデータを読み出すと，データ5が読み出されます（もう1回読み出すと今度はデータ4ですね）。

このように，後から入れたものが先に出てくることを，LIFOと呼びます。反対にFIFO（First In First Out）ということばもあります（入場券を買う人の列などがそうです。先に並んだ人が，先に切符を買えます）。

また，スタックにデータを入れることを「スタックに積む」といいますが，この感覚も理解されることと思います（スタックに積むことをpush，スタックから取り出すことをpullまたはpopといいます）。

2 スタックの実際

ではこのスタック，具体的にはどのようになっているのでしょうか。

実はスタックも，通常のアドレス空間に位置しています。そして，スタックの一番上を示すポインタ（アドレスを指し示すもの）が，データの入るべき（または出すべき）アドレスを指し示すのです。このポインタは，スタックポインタと呼ばれ，6809 CPUのレジスタの1つになっているのです（この連載ではA，B，CC，PC，X，Yに次いで7番目に登場したレジスタになりますね）。スタックポインタは，アドレスを指定するのだから，当然ながら16ビットのレジスタです（図4）。

図5を見てください。仮にスタックポインタの値が$7C00だとします。ここでDATA1という1バイトの値をスタックに入れると，今までのスタックの上，つまりスタックポインタの値である$7C00番地の上に（つまり$7BFF番地に）この値が入るのです（このように，スタックはアドレスの小さい方へと，メモリを利用していきます）。スタックの一番上のアドレスが1つ減ったので，スタックポインタの値も1つ減って$7BFFになります。この動作を式で表すなら，

S−1→S； r→（S）

ただし r：スタックに入れる値の入ったレジスタ
（ ）：カッコ内の値のアドレスの内容

という具合になります。

また，データを読み出すときは，スタックの一番上のデータから読みます。すなわち，スタックポインタで示されるアドレスの内容を読み出すのです。しかし，これだけでは不充分です。スタックの一番上のアドレスが変化したのですから，スタックポインタの値もそれに合わせて変更する必要があります。つまり，スタックポインタを＋1してやるわけです。

この動作を式で表すと，

（S）→r； S＋1→S

ただし r：スタックから出した値を入れるレジスタ
（ ）：カッコ内の値のアドレスの内容

になりますね。

図5を図2と比べると，スタックの考え方がそのまま具体化されているのがわかると思います。なお，図5（C）のようにスタックからデータを読み出しても，データはメモリ上に残ります。しかし，この値も新たにスタックにデータを積んだら消滅してしまうのです。まだメモリに残っているとはいっても，スタックポインタがそこより大きいアドレスを示していたら「お前はすでに死んでいる」ということになり，意味のないデータというわけです。

つまり，スタックポインタの示すアドレ

図4 スタックポインタ S

図5 スタックポインタとアドレス

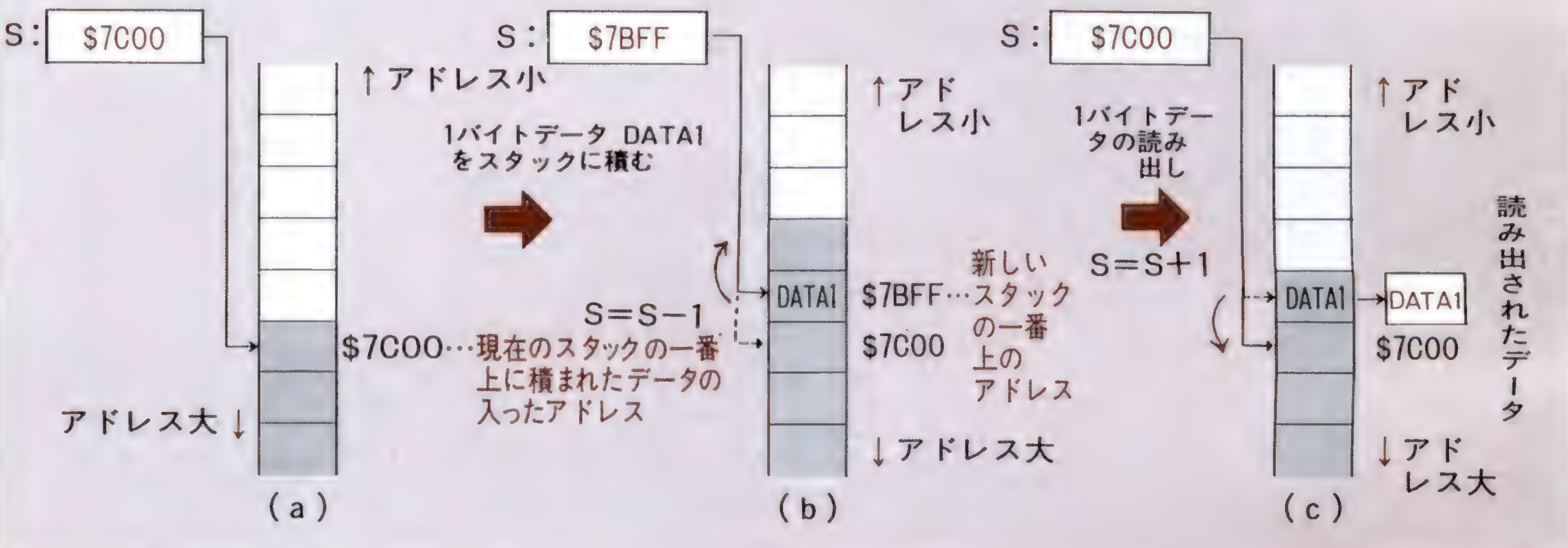


図6 データの破壊

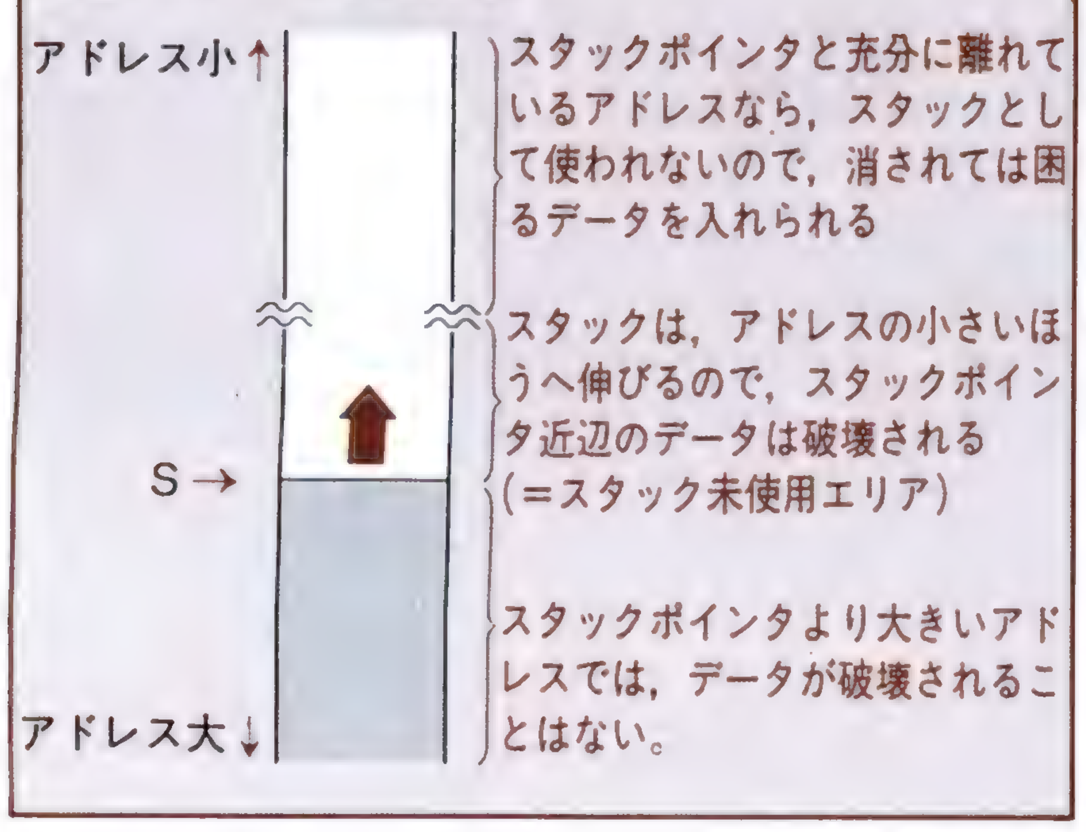
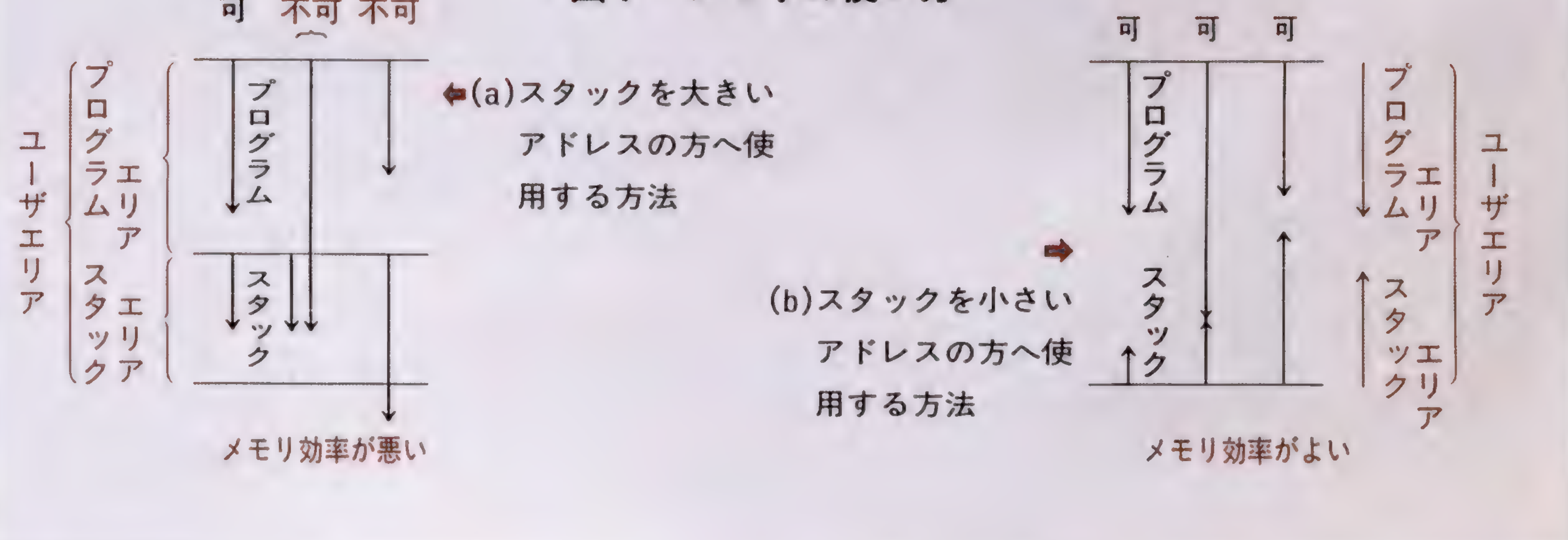


図7 メモリの使い方



スより小さいアドレスの内容は，ただのスタックの未使用エリアに過ぎないのです。といっても数十バイト程度を未使用スタックエリアとしてみておく必要はありますが，スタックにそれほど多くのデータを入れることはありません。スタックポインタから離れたアドレスには，保存が必要なデータを入れてもかまいません（図6）。

ところで，スタックがなぜ小さいアドレスの方へと使用されていくかというと，この方がメモリ利用効率がよいからです。これは図7を見てもらえばわかると思います。

3 スタックを使う命令

スタックのしくみがわかったところで実際にスタックを使用する命令をみてみましょう。

○サブルーチンコール

実はサブルーチンコール(JSR，BSRとRTS)はスタックを使う命令だったのです。

サブルーチンは，処理が終わったらサブルーチンを呼んだ所に帰ってこなくてはなりません。そのためには戻り先のアドレスを覚えておく必要があるのです。戻り先を記憶するのにスタックを使うわけです。

JSRや，BSRでは，まずPC（プログラムカウンタ）の値をJSRやBSRの命令長だけ増やします。このときPCは，サブルーチンの処理が終わったときに次に実行すべき命令の入ったアドレスを示しています。

次に，PCの値をスタックに入れます。PCは16ビットのレジスタなので，スタックの2バイトを使って収納されます。もちろん他の16ビットストアと同様に，上位バイトがアドレスの小さい方に，下位バイトがアドレスの大きい方に入るのです。

その後でジャンプ先のアドレスをPCに入れる（すなわちサブルーチンジャンプを行う）のです（図8）。

そうすると，RTSの動作のためには，スタックに積んでおいた戻り先のアドレスを取り出してPCに入れてやればよいことになります（図9）。

こうすれば，サブルーチンのネスティングもうまくいきます。ネスティングしているときのスタックの様子を図10に示します。

ところで，サブルーチン内で勝手にスタックの値を変化させたらどうなるでしょう。スタックポインタの値がサブルーチン開始時とリターン時で変わっていたら，RTSで戻るべきアドレスとは違った値をスタックから引っぱり出してきます。その結果，意図しないアドレスにジャンプすることになり，これはプログラムのりっぱな暴走です。

もちろん，スタックを使ってはいけないわけではありません。データを積む(push)のとデータを出す（pull）のが同じ回数（同じバイト数）になっていれば，スタックポインタはサブルーチンに入ったとき（エントリしたとき）と，リターン時で同じ値になるからです。

○PSHSとPULS

続いて，レジスタの内容をスタックに積む命令と，スタックから出す命令です。

まずレジスタの内容をスタックに積む（push）命令です。これは，

PSHS 〈レジスタリスト〉

の形で用います。オペランドに，スタックに積みたいレジスタ名をカンマで区切って並べるのです。並べる順は問いません。たとえば，

PSHS A

ならAレジスタのみをスタックに積み

図8 JSRの動作

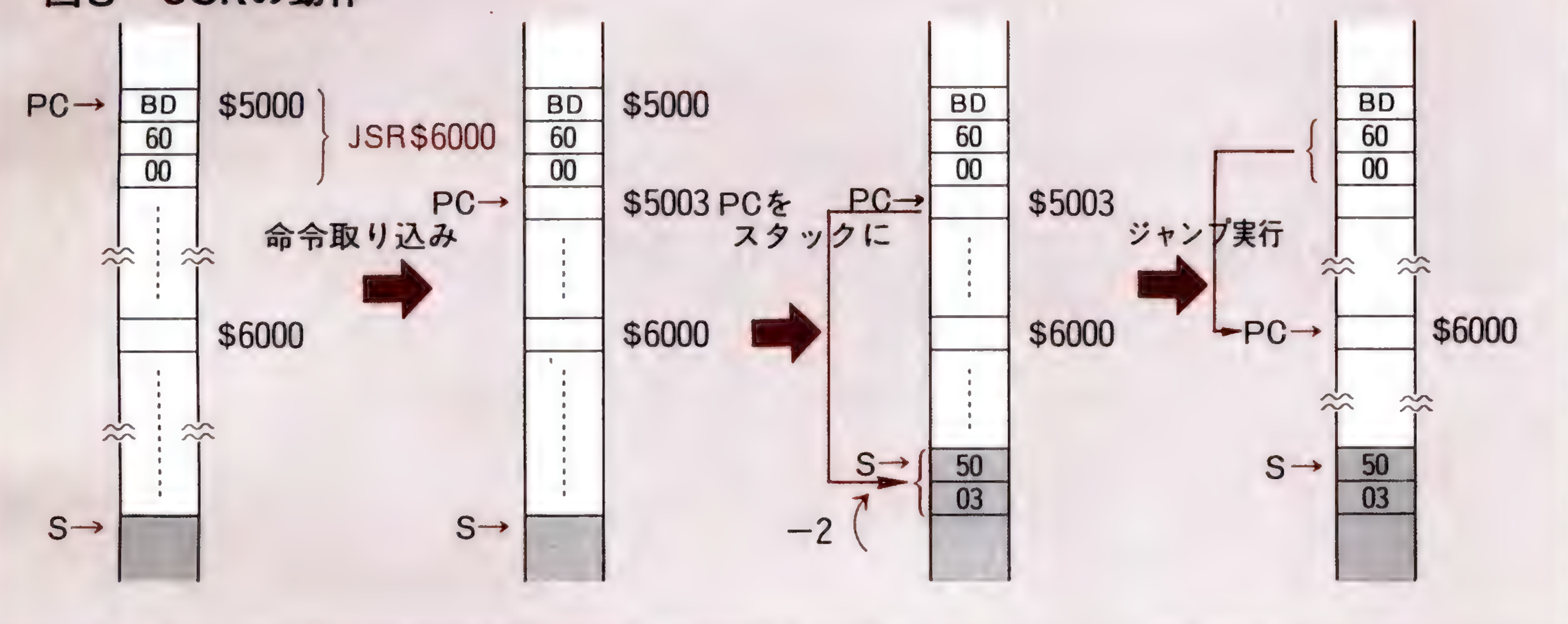


図9 RTSの動作

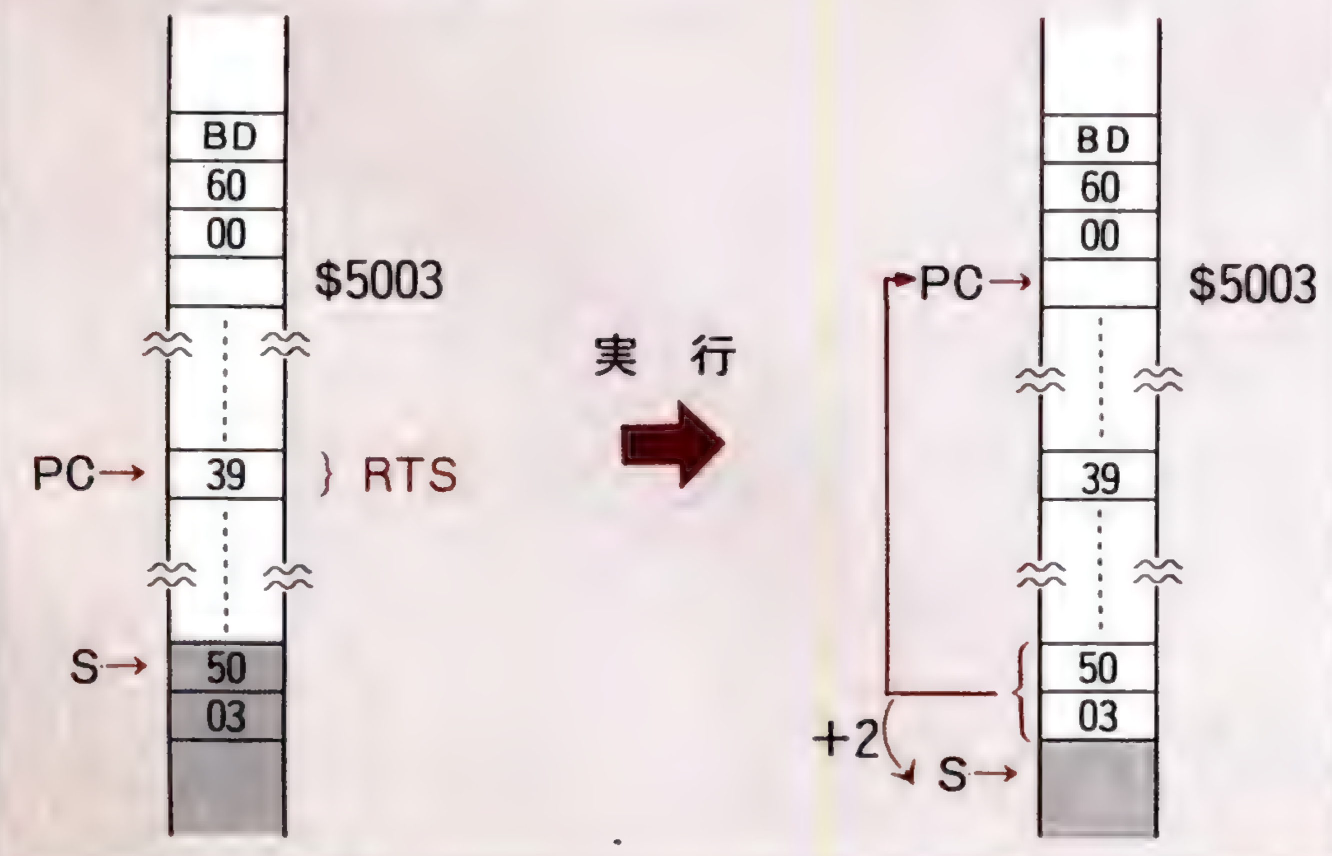
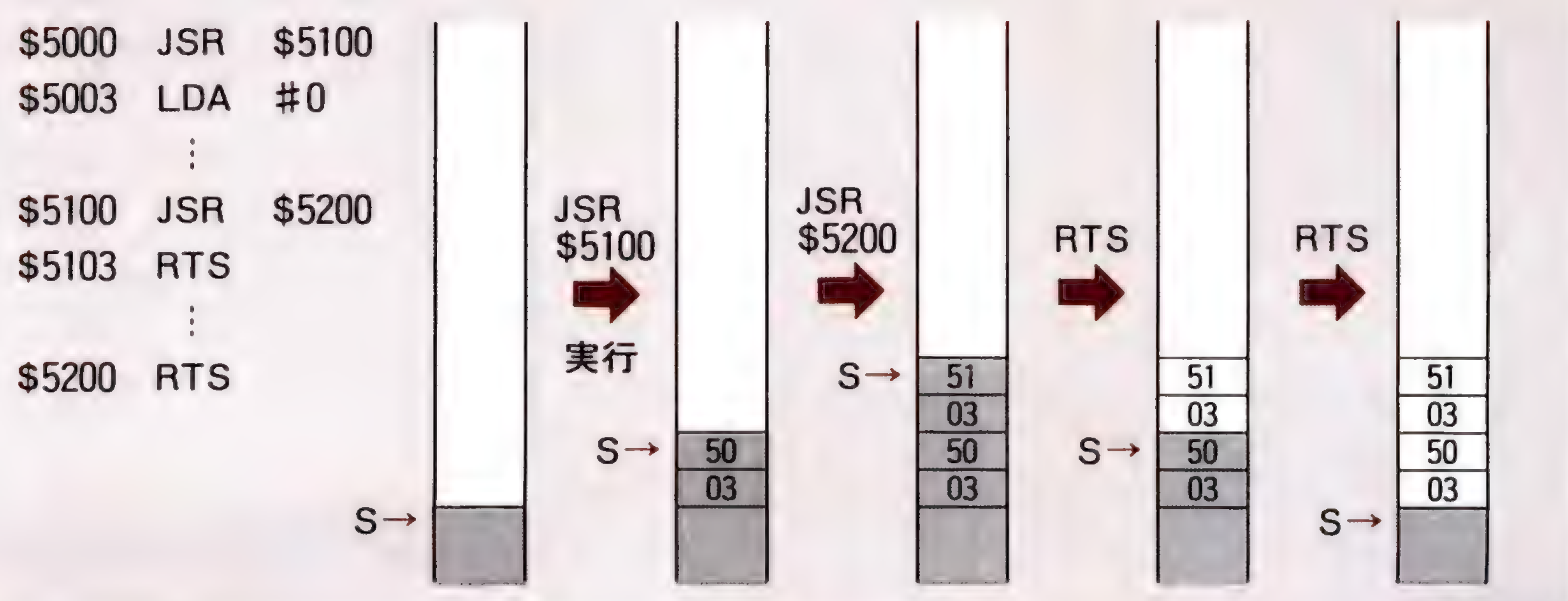


図10 ネスティングのようす



PSHS X, Y

ならXレジスタとYレジスタをスタックに積みます。16ビットレジスタは，スタックに2バイト積まれる点に注目してください（スタックポインタも－2されます）。

なお，スタックに積むことができるのは，スタックポインタ自身を除く全レジスタです。6809のレジスタは，未紹介のDP, U(それぞれ8ビット，16ビットのレジスタ）を含めて9個ありますから，8個のレジスタのなかから好きなだけ選べることになります。それに，A, Bの2つのレジスタという意味でDと記すのも可です。

また，複数のレジスタを指定したとき，どのレジスタが先に積まれるかの優先順位は，**図11**のようになります。

つまり

PSHS X, A, Y

と並べたところでX, A, Yの順にはならず，**図11**に示されるように，Y, X, Aの順になるのです（**図12**)。

ではPSHSの機械語コードを調べてみましょう。

命令表（**表1**）ではOPコードは$34，命令長が2バイトとなっています。これは，PSHSという命令を$34で示し，どのレジスタを指定するかを次の1バイトで示すからです。この2バイト目は，インデックスアドレッシングの2バイト目と同様に，ポストバイトと呼ばれますが，表す意味はまったく異なります。

PSHSのポストバイトは，各ビットが対応するレジスタを，指定するかしないかを，1と0で表しているのです（**図11**)。

たとえば，

PSHS CC, A, X, PC

は，

34 93 (=10010011(2))

の2バイトで示されます。

次はスタックからレジスタへ読み出す命令です。こちらは，

PULS 〈レジスタリスト〉

の形で用います。レジスタリストのフォーマットは，先のPSHSと同じで，

PULS A

ならスタックから1バイト読んで，Aレジスタに入れ，

PULS X, Y

ならX, Yレジスタにスタックからそれぞれ2バイトずつ読み出された値が入ります。複数のレジスタを指定した場合，pullされる順序はpushとちょうど逆になります（**図11**)。

機械語コードの方も命令表を見てわかるとおり1バイトのOPコード（$35）とレジスタを指定するポストバイトの計2バイトの命令となります。ポストバイトの形式はPSHSと同じです。たとえば，

PULS CC, A, X, PC

は，

35 93 (=10010011(2))

となります。

PULSの動作の例を**図13**にあげましたので，参考にしてください。

さて，PSHS／PULSの使用上の注意点をあげておきます。

まずフラグの変化です。PSHS／PULS命令は，「PULS CC」以外ではフラグは変化しません。また，前のサブルーチンコールの項でも触れましたが，1つのサブルーチン内ではスタックポインタの値が最初と最後で一致しなくてはなりません。そのためPSHSした分は必ずPULSを行って（あるいはそれに相当する処理：次項スタックポインタに関する命令参照）ください。何もスタック

図11 レジスタの優先順位

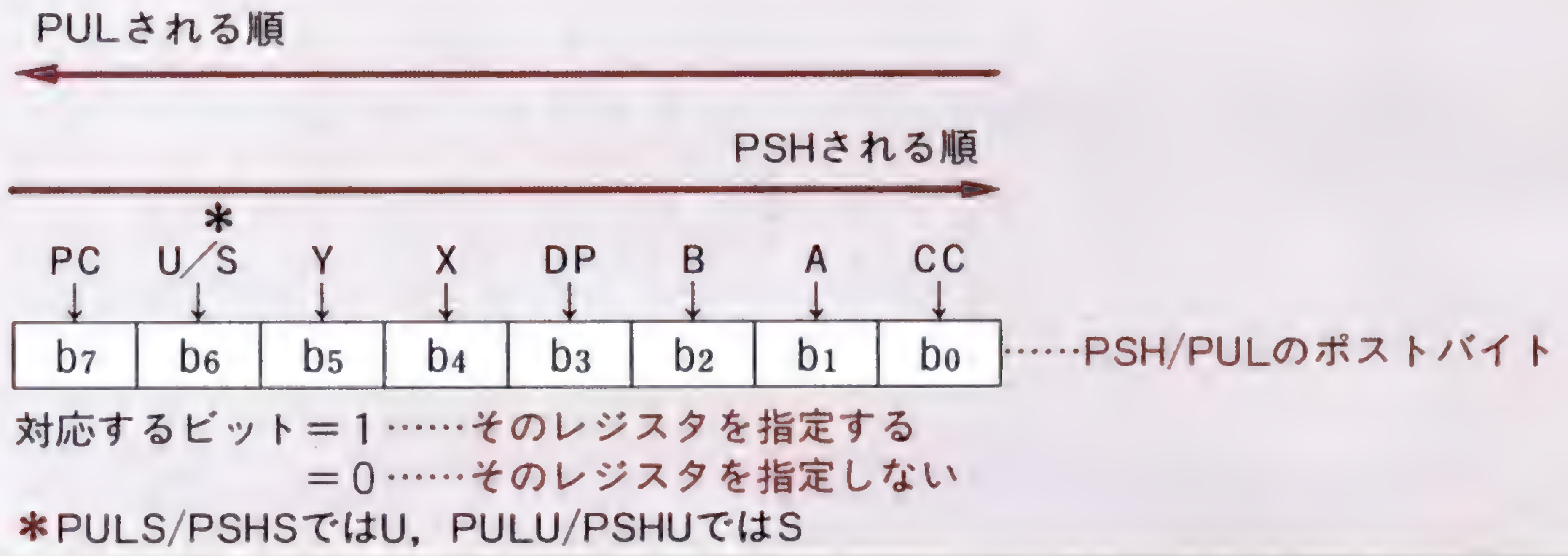


図12 PSHSの動作

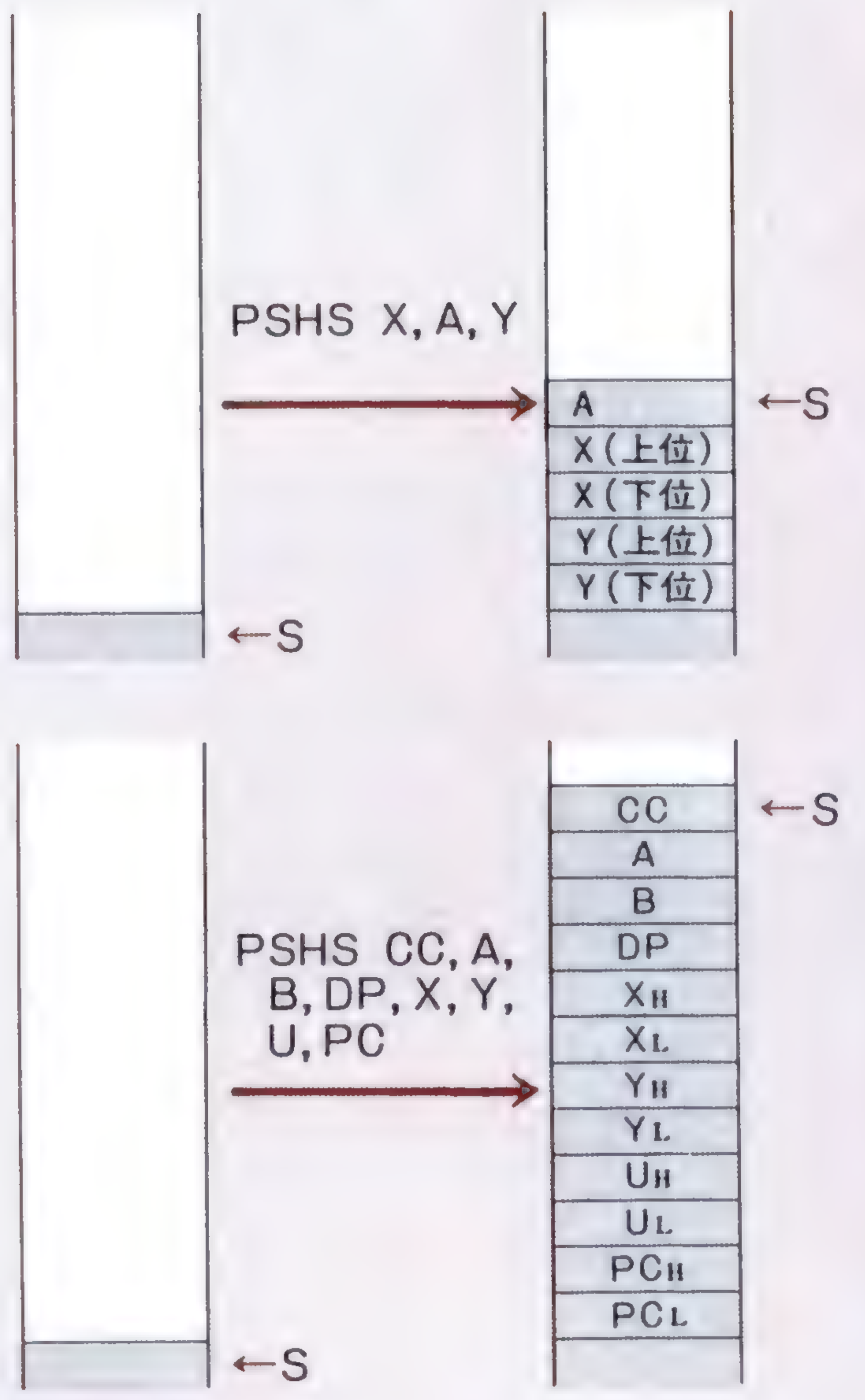
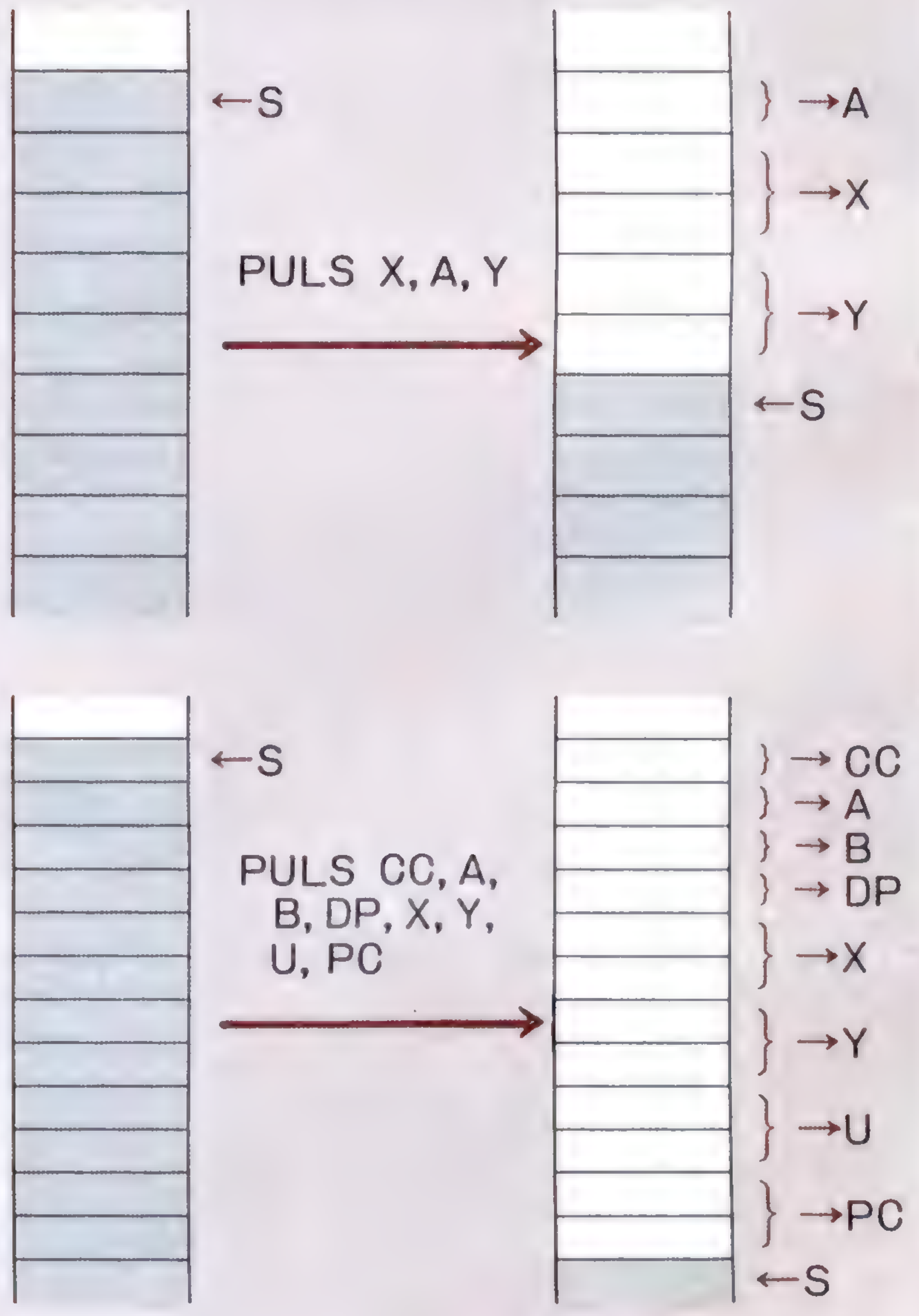


図13 PULSの動作



に積まない状態でPULSを行うのは，もし普通にRTSでサブルーチンから戻りたいならマズいことになります。

しかし，このことはわかっていても，プログラムを作っていると，「このレジスタの値をスタックにとっておこう」と思ってPSHS命令を使い，後でPULSを行うのを忘れることがよくあります。そうするとプログラムが暴走してバグを知らせてくれますので，そのときは**念入りに**スタックを何バイト使用したか１命令ずつ追って，チェックしてみてください（念入りに行うのですよ。このあいだ「チェックしたのに暴走する」とのたもうた友人のプログラムをチェックしたら，やはりスタックレベルが合ってませんでした)。特に，条件分岐で必要なPULSを飛びこしてしまうとか，忘れるなどもよくあるミスなので気を付けましょう。

では具体的にスタック操作を行うプログラム例を示します。

リスト１は３重ループをＡレジスタだけ使って実行した例です。ループのカウンタを次のループの始まる前にスタックに入れて値を退避させています。プログラムリストの右側にスタック使用中のバイト数を示しました。分岐先とスタックレベルが合っている点に注目してください。

また，レジスタの値をサブルーチン側で変更されては困るときにも，サブルーチンの入口でPSHS，最後にPULSを行うことがあります（**リスト２**)。

表１　命令表（スタック関係を抜すい）

命令		INH ACCM REG			DIRECT			EXTND			IMMED			INDEX①			RELATIVE			動作	7	6	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		E	F	H	I	N	Z	V	C
CMP	CMPS				11 9C	7	3	11 BC	8	4	11 8C	5	4	11 AC	7+	3+				Compare M：M+1 from S	•	•	•	•	↕	↕	↕	↕
	CMPU				11 93	7	3	11 B3	8	4	11 83	5	4	11 A3	7+	3+				Compare M：M+1 from U	•	•	•	•	↕	↕	↕	↕
EXG	R1, R2	1E	8	2																R1→R2②	(			⑤				)
LD	LDS				10 DE	6	3	10 FE	7	4	10 CE	4	4	10 EE	6+	3+				M：M+1→S	•	•	•	•	↕	↕	R	•
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				M：M+1→U	•	•	•	•	↕	↕	R	•
LEA	LEAS													32	4+	2+				EA③→S	•	•	•	•	•	•	•	•
	LEAU													33	4+	2+				EA③→U	•	•	•	•	•	•	•	•
PSH	PSHS	34	5+④	2																Push Registers on S Stack	•	•	•	•	•	•	•	•
	PSHU	36	5+④	2																Push Registers on U Stack	•	•	•	•	•	•	•	•
PUL	PULS	35	5+④	2																Pull Registers from S Stack	(			⑤				)
	PULU	37	5+④	2																Pull Registers from U Stack	(			⑤				)
ST	STS				10 DF	6	3	10 FF	7	4				10 EF	6+	3+				S→M：M+1	•	•	•	•	↕	↕	R	•
	STU				DF	5	2	FF	6	3				EF	5+	2+				U→M：M+1	•	•	•	•	↕	↕	R	•
TFR	R1, R2	1F	6	2																R1→R2②	(			⑤				)

（記号および略称の説明）

(1) オペレーション記号
　→＝転送方向

(2) 略称記号
　OP＝オペレーションコード（16進表示）
　～＝CPUのサイクル数
　＃＝命令語のバイト数

(3) メモリとアドレス形式
　M　＝記憶番地
　M+1　＝記憶番地に１を加えた記憶番地
　IMM　＝イミディエイト値
　INH　＝インヒレントアドレス形式
　ACCM　＝アキュムレータアドレス形式
　REG　＝レジスタアドレス形式
　DIRECT　＝ダイレクトアドレス形式
　EXTND　＝エクステンドアドレス形式
　IMMED　＝イミディエイトアドレス形式
　INDEX　＝インデックスアドレス形式
　RELATIVE＝リラティブアドレス形式

(4) コンディションコードレジスタのビット０～ビット７の意味
　E（b_7）＝エンタイアフラグ(全レジスタを復帰するか否か)
　F（b_6）＝FIRQマスクフラグ
　H（b_5）＝ビット３からビット４への桁上げフラグ
　I（b_4）＝IRQマスクフラグ
　N（b_3）＝負のフラグ
　Z（b_2）＝ゼロのフラグ
　V（b_1）＝２の補数の場合のオーバフローフラグ
　C（b_0）＝b_7から桁上げまたはb_7への桁借りフラグ

(5) コンディションコードレジスタの内容変化を示す記号
　R＝常にリセットされる。
　↕＝条件が満たされたときセットされ，そうでないときは，リセットされる。
　•＝その命令では変化しない。

注)

①＝インデックストモードのときは，この表に示すマシンサイクル数およびバイト数にOh／FM７月号，P.130のポストバイト一覧表に示す値を加えた値が必要なマシンサイクル数およびバイト数になる。

②＝EXG命令とTFR命令のR_1とR_2は８ビットレジスタどうしか16ビットレジスタどうしである。ただし，TFR命令ではR_1=16ビット，R_2=８ビットの転送は許可する。
　{８ビットレジスタ：A, B, CC, DP
　{16ビットレジスタ：X, Y, U, S, D, PC

③＝EAは実効アドレス(EFFECTIVE ADDRESS)である。

④＝PSH/PUL命令のサイクル数
　表に示されているマシンサイクル数に，PSH/PULされるレジスタのバイト数を加える。

⑤＝オペランドにCCが指定された場合（ただし，TFRの命令のときは転送先にCCが指定された場合）には命令実行の結果にしたがってセットされる。
　その他の場合は変化しない。

4 スタックポインタに関する命令

ほとんどの8ビットCPUでは，スタック操作の命令（jump subroutineやpush／pull)以外に他のレジスタの値をスタックポインタに入れる，およびその逆の命令しか持っていません。普通はこれで充分ですが，6809の場合，スタックポインタはインデックスレジスタ(XやY)に対する命令と同一のものを備えています。つまり，インデックスレジスタの機能を持っているのです。そのため，LD, LEA, ST, CMPといったロード，ストア，比較命令はもちろん，インデックスアドレッシングにXやYでなくSを指定することができるのです。

もちろんSレジスタをインデックスレジスタのつもりで使い，やたらに値を書き換えると，スタックポインタとしての機能が果たせなくなる（→暴走につながる）ので不都合です。これらの機能は，スタックを使いやすくするためにあると思ってください。

○インデックスレジスタとしての使用

インデックスアドレッシングで，インデックスレジスタとしてSを指定できます。用法は，XやYのかわりにSを用いて

```
LDA  ,S
ADDD 2,S
```

などとすればよいのです。機械語コードの方は，7月号，130ページの表1でポストバイトの'RR'のところに'11'を入れたものを選ぶだけの違いです。

スタックに積んだ値を読み出すのに，PULSを使うと，その値はスタックから消えてしまいますが，たとえば

```
LDA  , S
```

のようにインデックスアドレッシングで読み出すと，スタックに積んだまま何度でも読むことができます。また，PULSは読み出した値をレジスタに入れることしかできませんが，インデックスアドレッシングでは命令のオペランドとしてスタックの値を使えるのです。

たとえば，

```
CMPA ,S
```

とすればスタックの一番上の1バイトとAレジスタの値の比較が行えます。さらに

```
ADDA ,S+
```

とすれば，スタックの一番上の値とAを加算した後，スタックの一番上のデータを捨てる（スタックポインタが1つ増えたのだからPULSを行ったのと同じです）動作まで行えます(FORTHを知っている人，6809っていいでしょ！）。

またこれを利用すればAレジスタとBレジスタの加算を（結果：Aに入れる），

```
PSHS B
ADDA ,S+
```

とスタックをちょっと利用するだけで行えます。

なお，

```
PSHS A ←→ STA ,-S
PULS X ←→ LDX ,S++
        ⋮
```

のように，PSHSと同じ動作をストア命令のインデックスアドレッシング プリデクリメントを使って，またPULSと同じ動作をロード命令のインデックスアドレッシング ポストインクリメントを使っても行えます。ただしフラグの変化は異なります（PSHS, PULSでは変化なし，LD, STではZ, N, Vが変わる)。

オートインクリメント／デクリメントでインクリメントがインデックスアドレッシングのあと＋，オートデクリメントはインデックスアドレッシングの前に－されるのは，スタックの動作と対応させるためだったのです。

○スタックポインタへのロード（LDS)

スタックポインタに値をロードする命令です。動作は（アドレッシングモードも含めて）LDXやLDYと同様です。

この命令はCPUのリセットシーケンス(要するにCPU動作の一番最初)の後，スタックを設定するため以外に使うことは，まずありません。スタックエリアを移動したいときに使うことがありますが，この場合，スタックに積まれたデータもいっしょに移してやる必要があります。

■リスト1■

```
10 FOR I=1 TO 100
20   FOR J=1 TO 100
30     FOR K=1 TO 100
40     NEXT
50   NEXT
60 NEXT
70 END
```

➡

```
                        スタックレベル(各命令実行前)
       LDA    #1        0
LOOP1  PSHS   A         0
       LDA    #1        1
LOOP2  PSHS   A         1
       LDA    #1        2
LOOP3  INCA             2
       CMPA   #100      2
       BLS    LOOP3     2
       PULS   A         2
       INCA             1
       CMPA   #100      1
       BLS    LOOP2     1
       PULS   A         1
       INCA             0
       CMPA   #100      0
       BLS    LOOP1     0
       RTS              0
       END
```

■リスト2■

```
SUB1  PSHS  A,B,X       ……レジスタを3つスタックに
        ⋮
        ⋮               } サブルーチン内の処理
        ⋮
      PULS  A,B,X,PC    ……PULS A, B, XとRTS(≒PULS PC)
                          を合わせるとこうなる
```

もちろんF-BASICでは立ち上げたときにスタックポインタの設定は行われている(図14)ので,再設定不要です(というよりやってはいけません)。

○スタックポインタの値のストア(STS)

スタックポインタの値のストア命令はSTSで,動作はSTXやSTYと同様です。

この命令を使うとしたらプログラムのデバッグでスタックポインタの値を知りたいときくらいでしょう。スタックの動作に不審を感じたら1命令入れてみてください。

○スタックポインタの値の比較(CMPS)

動作・用法はCMPX, CMPYと同じです。スタックがやたらに大きくなっては困るとき,チェックに使えます(たとえばスタックを使いすぎ,プログラムエリアを潰す恐れがあるときなど)。

○実効アドレスロード(LEAS)

LEAX, LEAYのスタックポインタ版です。ただし,フラグが変化しない点に注意してください。この命令は,スタックポインタ自体を操作する命令で,よく使われます。たとえば,スタックから2バイトの値を捨てたいとき(スタックにデータを入れたものの,それを無効にしたい場合など),

LEAS 2,S

とすれば,スタックの一番上の2バイトが,PULSを使わずに,放棄されたことになります(図15)。

また,オフセットかマイナスのインデックスアドレッシングを用いて,たとえば

LEAS −10,S

とすれば,10バイトpushしたのとスタックポインタの値が同じ,すなわち10バイト分有効データが入るスペースがスタック上にできたことになります。ここはワーク(作業)エリアとして利用できるのです。もちろんこのエリアはスタックポインタをインデックスレジスタとして使うインデックスアドレッシングで参照(値の読み出しや,書き込みなど)するのです。この場合,このエリアの使用が済んだとき,

LEAS 10,S

でワークエリアを処分します(スタック使用上の注意を思い出してください)。

こうしてワークエリアを確保すると,再帰的プログラムが簡単に作れます。これらについては,連載の後半で詳しく述べたいと思っています(このあたりが,6809のオイシイところなのです。6809なら動的局所変数は,こうも簡単に扱えるんですよ! Cを知っているみなさん)。

さて,ここまでの知識をもとに,前回の

図14 スタックエリア

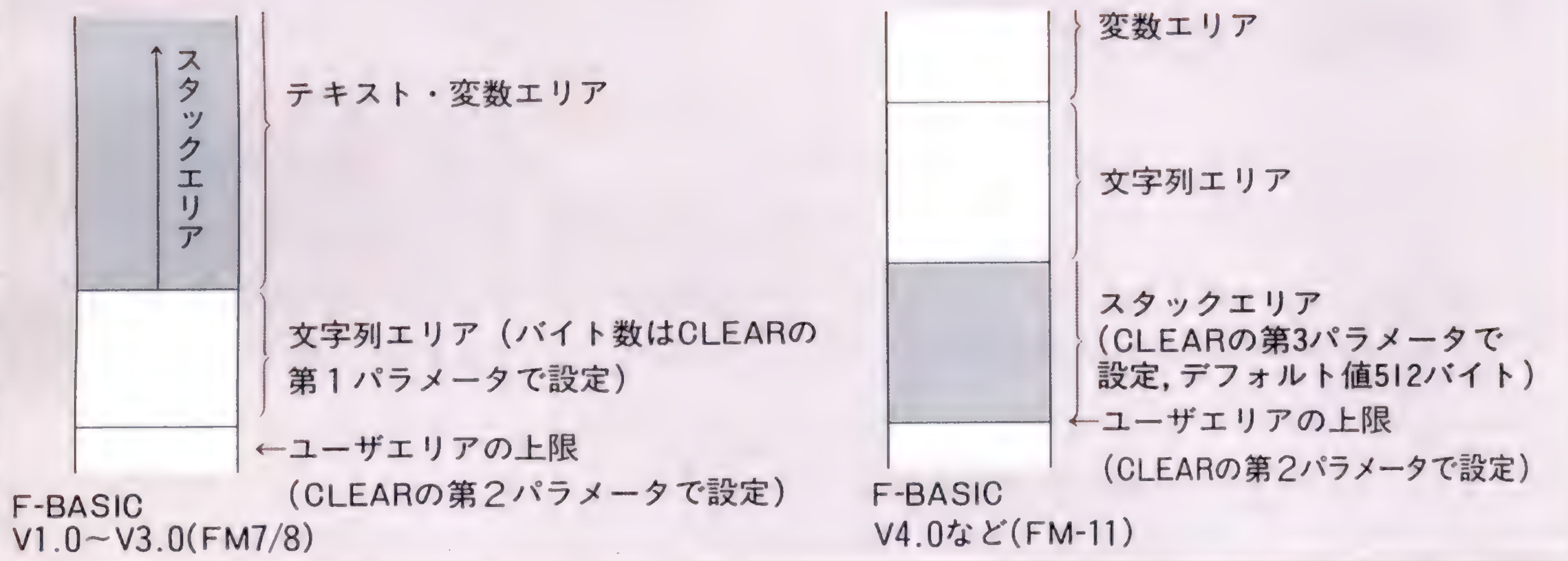
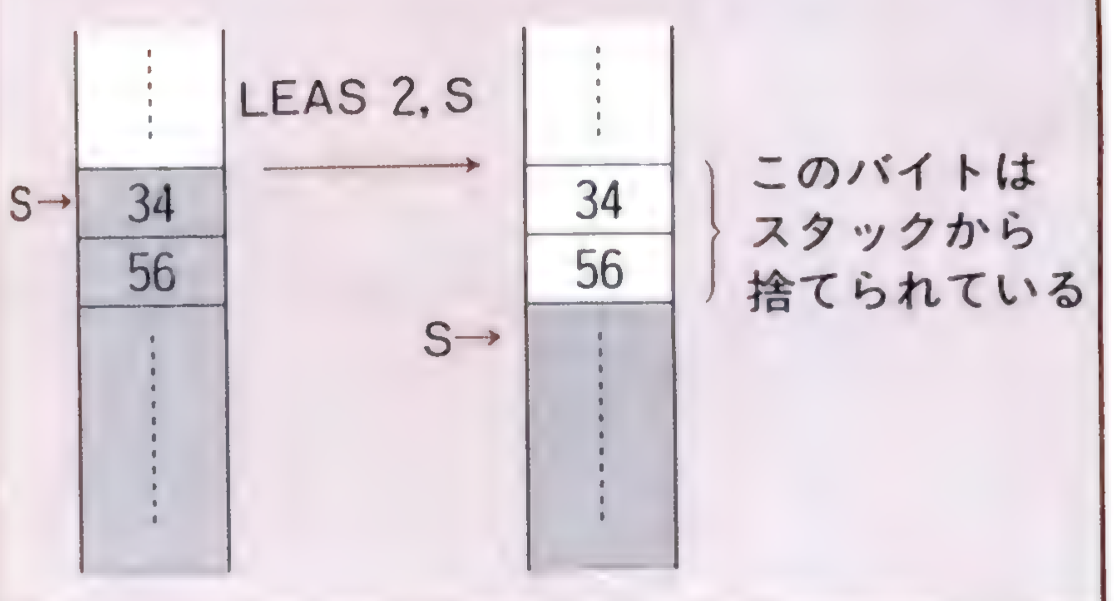


図15 PLUSを使わずに
スタックのデータを捨てる



■リスト3■

```
                              *      get max value routine
5000                                  ORG      $5000
5000 34    10                         PSHS     X          ;backup X
5002 AE    02                         LDX      2,X        ;load ﾊｲﾚﾂ top addr.
5004 34    10                         PSHS     X          ;stack top is ﾊｲﾚﾂ-top-addr.
5006 30    89 07CE                    LEAX     1998,X     ;load ﾊｲﾚﾂ end addr.
500A 31    84                         LEAY     ,X         ;load ﾊｲﾚﾂ end addr.
500C EC    A4                         LDD      ,Y         ;load ｶﾘﾉ ｻｲﾀﾞｲﾁ(Y:ｿﾉ address)
500E 10A3 83                  CMPLOP  CMPD     ,--X       ;compare with another data
5011 2E    04    5017                 BGT      CMPNXT     ;D>,X.
5013 31    84                         LEAY     ,X         ;change ｶﾘﾉ ｻｲﾀﾞｲﾁﾉ address
5015 EC    A4                         LDD      ,Y         ;load ｶﾘﾉ ｻｲﾀﾞｲﾁ(Y:ｿﾉ address)
5017 AC    E4                 CMPNXT  CMPX     ,S         ;ﾊｲﾚﾂ top?
5019 22    F3    500E                 BHI      CMPLOP     ;ﾏﾀﾞﾀﾞﾖ.-> goto CMPLOP
501B 34    20                         PSHS     Y          ;ｻｲﾀﾞｲﾁﾉ addressｦ D-reg ﾆ.     } TFR Y,Dを使うとよい
501D 35    06                         PULS     D          ;  (ﾎﾝﾗｲﾊ TFR Y,D ｦ ﾂｶｳ)       }
501F A3    E1                         SUBD     ,S++       ;D-(stack top)-> D
5021 44                               LSRA                ;ﾌｺﾞｳﾅｼ ﾊ LSR｡(D/2=ﾊｲﾚﾂﾉ ﾅﾝﾊﾞﾝﾒｶ)  } D=D/2
5022 56                               RORB                ;ｼﾞｮｳｲ､ｶｲﾉ ｼﾞｭﾝ｡(shift rightﾃﾞｽ)   }
5023 35    10                         PULS     X          ;X:ﾋｷｽｳaddress
5025 ED    02                         STD      2,X        ;ｱﾀｲｦ BASICﾆ ﾜﾀｽ
5027 39                               RTS
                                      END
```

リスト7を改良してみましょう。**リスト3**を見てください。ずいぶんとエレガントになったと思いませんか。

5 ユーザスタックポインタU

6809は，スタックポインタがもう1つあります。といっても，それは**ユーザスタックポインタ**（Uで表す：**図16**）といって，JSRなどで使用されることはありません。

つまり，値を変えても，暴走の直接の原因にはならないものです。もちろんスタックポインタと呼ぶからには，push／pull命令が使えます。なお，ユーザスタックポインタに対し，Sを**システム（ハードウェア）スタックポインタ**と呼びます。Uは，Sと同様，インデックスレジスタの機能を持っているので，ユーザスタックポインタとしてよりも，インデックスレジスタUとして使われることが多いようです。普通，スタックは，1つあれば充分です。

○ PSHU

スタックポインタとしてUを使うとき，スタックにデータを入れる命令です。動作は用いるスタックポインタが異なる点，スタックに積めるレジスタがUを除く8個となっている点以外，PSHSと同じです。レジスタの優先順位は，**図11**を参照してください。

なお，PSHS, PSHUはそれぞれPSH命令のS版，U版というわけです。

○ PULU

こちらもPULSのUレジスタ版です。

○ LDU, STU, CMPU, LEAU

当然これらの命令も使えます。動作はLDS, STS, CMPS, LEASと同じです。

ただしUレジスタはインデックスレジスタとして用いることが多く，これらの命令の用途は，Sレジスタの場合と変わってきます。つまりLDX, STX, CMPX, LEAXと同じように用いるということです。LEAUとLEAXのフラグの変化の違いだけは，よく頭に入れてください。

第7章 特殊なアドレッシング

第1章でイミディエイト，およびエクステンドアドレッシングを，第2章でインデックスアドレッシングを扱いましたが，この章では6809のアドレッシングモードの残りすべてについて，話を進めます。どれも6809独特のもので，これらを使いこなすのが，6809を生かすプログラムなのです。

1 ダイレクトアドレッシング

ダイレクトアドレッシングは，アドレス16ビットのうち下位8ビットだけをオペランドで指定するものです。残りの上位8ビットは「**ダイレクトページレジスタ（DP）**」というレジスタ（**図17**）の値になります（**図18**）。なお，アドレス上位8ビットが同じアドレスの範囲をページと呼ぶことがあります。

アセンブリ言語での記述は，

LDA　$30……（a）

または，

LDA　<$30……（b）

とします。（a）の形ではエクステンドアドレッシングと同じですが，通常アセンブラでは，アドレスの上位8ビットが0のときはダイレクト，そうでないときはエクステンドと扱うからです（普通DPは$00にして使います）。

ダイレクトアドレッシングであることをはっきりさせたいときは（b）のように「<」を付けます。

もちろんDPの値は，後述の方法で変更できるので，たとえばDPの値が＄FDなら，

LDA　$FD05

をダイレクトアドレッシングにしてもよいわけです。ちゃんとした6809用のアセンブラなら，

SETDP　〈DPの値〉

という疑似命令を実行しておけば，アドレスの上位バイトが，この疑似命令で指定した**仮想的なDPの値**（アセンブラ実行時のDPの値）と一致したときにダイレクトアドレッシング，一致しないときにエクステ

図16　ユーザスタックポインタU

図17　ダイレクトページレジスタDP

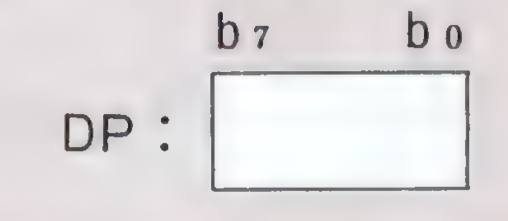
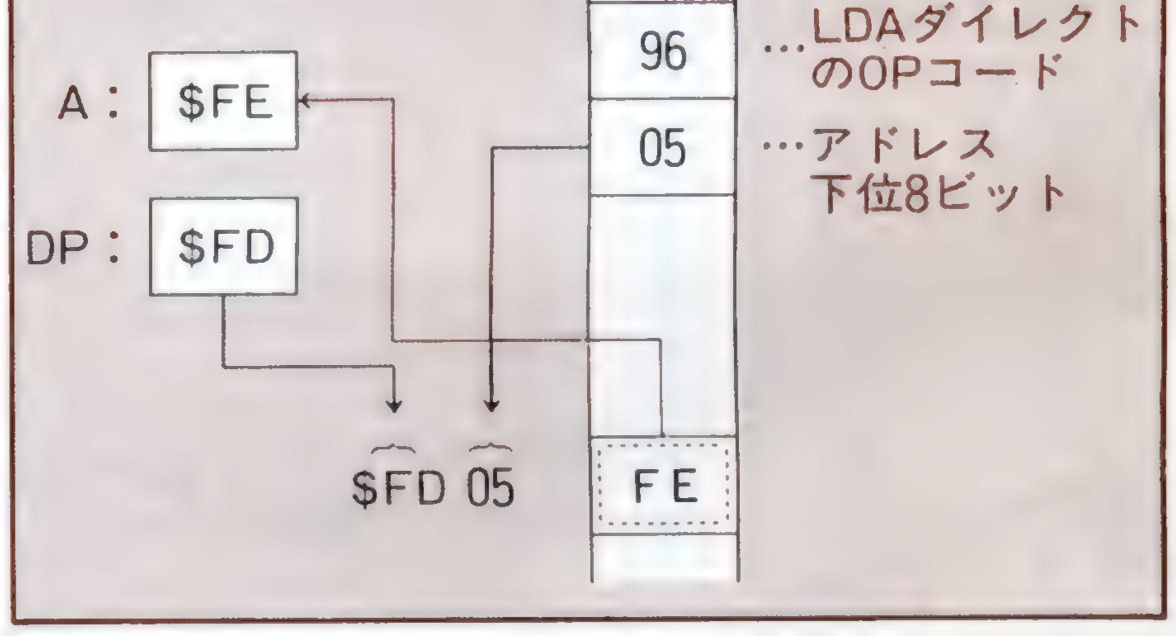


図18　DPの値

例　LDA　$05



■リスト4■

```
0000 96    20              LDA    $20    …ダイレクト（アドレス上位＝＄00だから）
0002 B6    FD05            LDA    $FD05  …エクステンド（アドレス上位≠$00）
0005 96    20              LDA    <$20   …ダイレクト
           00FD            SETDP  $FD    …ダイレクトページを$FDに（$FDXXはダイレクトにできる）
0007 B6    0020            LDA    $20    …エクステンド（アドレス上位＝＄00≠$FD）
000A 96    05              LDA    $FD05  …ダイレクト（アドレス上位＝＄FD）
****WARNING   002--00000
000C 96    20              LDA    <$20   …ダイレクト（強制的に。そのため上の行に
                                           注意のメッセージ）
```

ンドアドレッシングと自動的に判定してくれます。なおSETDPの疑似命令は，あくまで疑似命令ですので，この命令は**実際のDPの値**を設定するものではありません。**リスト4**にエクステンドアドレッシングとダイレクトアドレッシングの判定の様子を示します。

次に機械語コードの求め方です。

まずOPコードは命令表のダイレクトのものを用います。たとえば，LDAなら＄96，STAなら＄97です。

また，次の1バイトに対象アドレスの下位8ビットを付けます。つまり，計2バイト（OPコードが2バイトのものは3バイト）の命令になります(**図19**)。このため，エクステンドアドレッシングより1バイト短くなります（実行時間も1サイクル速くなります)。

たとえばLDA　＜＄97は

96　97

の2バイトになるわけです。

○TFRとEXG

DPの設定は，前出のPUL命令でスタックから1バイト取ってくるか，これから述べる**TFR/EXG**命令でしか行えません。

TFR（Trans Fer Register）はレジスタの内容を他のレジスタに移す命令で，

TFR　R_1，R_2

（R_1，R_2はともに8ビットまたは16ビットのレジスタ。ただしR_1が16ビット，R_2が8ビットは可。）

の形で用い，動作は，

$R_1 \rightarrow R_2$

の転送を行います。たとえば

TFR　D，X

はDレジスタの値をXレジスタに入れ，

TFR　A，DP

はAレジスタの値をDPレジスタに入れます。

機械語コードは，OPコード＄1F，それに転送元・転送先レジスタを指定するポストバイトの計2バイトとなります。

ポストバイトは**図20**のように，上位4ビットでR_1（転送元）を，下位4ビットでR_2(転送先)を指定します。すなわち16進数で記した場合，上の桁が転送元を，下の桁が転送先を表すのです。

たとえば，

TFR　D，X　→ 1F　01

TFR　A，DP → 1F　8B

となります。

EXG（EXchanGe register）は2つのレジスタの内容を入れ換える命令で，

EXG　R_1，R_2

（R_1，R_2はともに8ビット，またはともに16ビットのレジスタ）

の形で用い，R_1とR_2のレジスタの内容を入れ換えます。たとえば

EXG　A，B

ではAレジスタとBレジスタの内容を交換します。

機械語コードは，OPコードが＄1E，それにTFRと同様R_1とR_2の指定を行うポストバイトで構成されます。したがって，

EXG　D，X　→ 1E　01

EXG　A，DP → 1E　8B

という具合になります。

2 インデックスアドレッシング・アキュムレータオフセット

続いて，インデックスアドレッシングのバリエーションの1つ，アキュムレータオフセットです。

これはインデックスアドレッシングでオフセットにアキュムレータ（AorBorD）の値を用いるモードです。

アセンブリ言語は，定数オフセットの定数のかわりにアキュムレータ名を書いて，

LDX　A，Y

STA　B，X

LEAX　D，X

などと用います。もちろんA，Bは8ビット符号付きオフセットとして，Dは16ビット符号付きオフセット（もちろん2章で述べたように符号なしと見ることもできます）として処理されます。

次に機械語コードですが，インデックスアドレッシングのOPコードと1バイトのポストバイトの計2，または3バイトの命

図19　ダイレクトアドレッシングの命令長

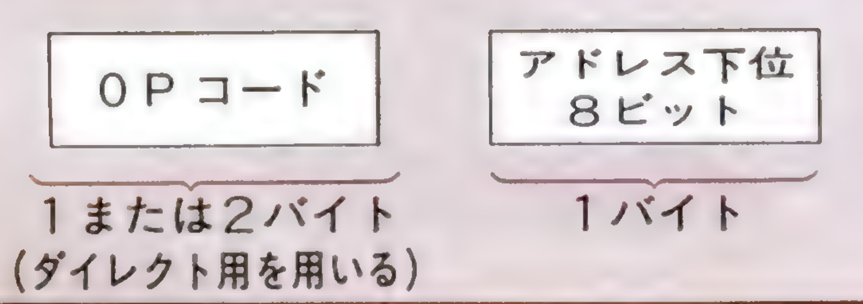


図21　インデックスアドレッシング　アキュムレータオフセットの命令長

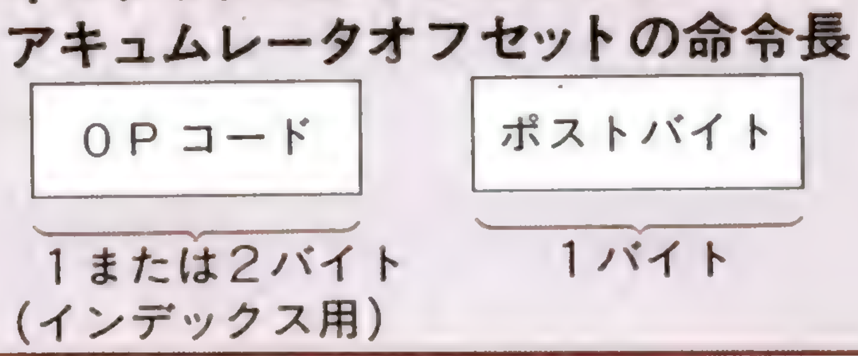
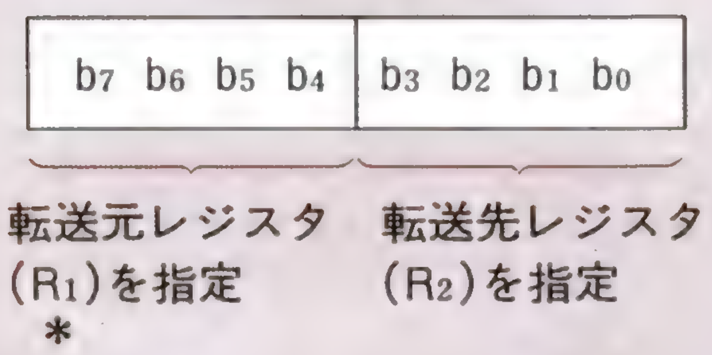


表2　アキュムレータオフセットのポストバイト

モード ＼ インデックスレジスタ	X	Y	U	S
A，　R	86	A6	C6	E6
B，　R	85	A5	C5	E5
D，　R	8B	AB	CB	EB

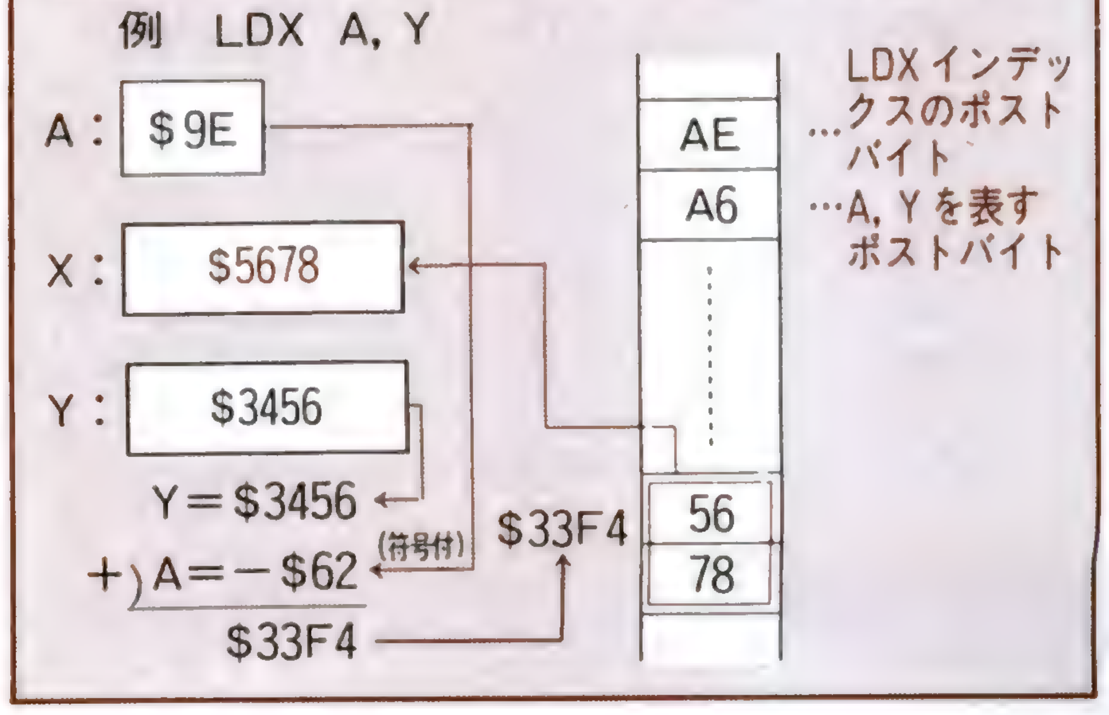
図20　TFR/EXGのポストバイト



＊レジスタ指定の4ビット

(a) 16ビットレジスタ	(b) 8ビットレジスタ
D……0000（＄0）	A……1000（＄8）
X……0001（＄1）	B……1001（＄9）
Y……0010（＄2）	CC……1010（＄A）
U……0011（＄3）	DP……1011（＄B）
S……0100（＄4）	
PC……0101（＄5）	

図22　アキュムレータオフセットの動作



■リスト5■

```
                  ⋮
180 ´TOHEX  LDX    #HEXTBL
190 ´       LDB    A,X
200 ´       RTS
210 ´HEXTBL FCC    ´0123456789ABCDEF´…変換用
                  ⋮                 データテーブル
```

■リスト6■

(a)

```
5000                        ORG    $5000
5000         0001    PARAM  RMB    1
5001 B6      5000           LDA    PARAM  ←ここがエントリアドレス
5004 8D      04             BSR    TOHEX    (＝$5001)
5006 F7      5000           STB    PARAM
5009 39                     RTS
                     *
500A 8E      5010    TOHEX  LDX    #HEXTBL
500D E6      86             LDB    A,X
500F 39                     RTS
5010         30      HEXTBL FCC    ´0123456789ABCDEF´
5011         31
5012         32
5013         33
5014         34
5015         35
5016         36
5017         37   変換のための
5018         38   データテーブル
5019         39
501A         41
501B         42
501C         43
501D         44
501E         45
501F         46
                     *
                            END
```

(b) (a)をテストするBASICプログラム(0～15を，0～'F'に)

```
10 FOR I=0 TO 15
20 POKE &H5000,I              …パラメータ渡し
30 EXEC &H5001                …コード変換
40 PRINTCHR$(PEEK(&H5000))    …ASCIIコードを文字に変換
50 NEXT
60 END
```

■リスト7■

```
100 ´        LDX    #JMPTBL
110 ´        ASLA
120 ´        LDX    A,X      …ジャンプ先のアドレス値をロード ┐ JMP〔A,X〕も可
130 ´        JMP    ,X       …Xの値のアドレスにジャンプ     ┘
140 ´JMPTBL  FDB    SHORI0
150 ´        FDB    SHORI1
160 ´        FDB    SHORI2
170 ´        FDB    SHORI3
180 ´        FDB    SHORI4
```

令になります(**図21**)。

ポストバイトは，7月号P130の表1で調べるのですが，アキュムレータオフセットの項のみ抜き出したものを**表2**に示します。動作例は，**図22**を参照してください。

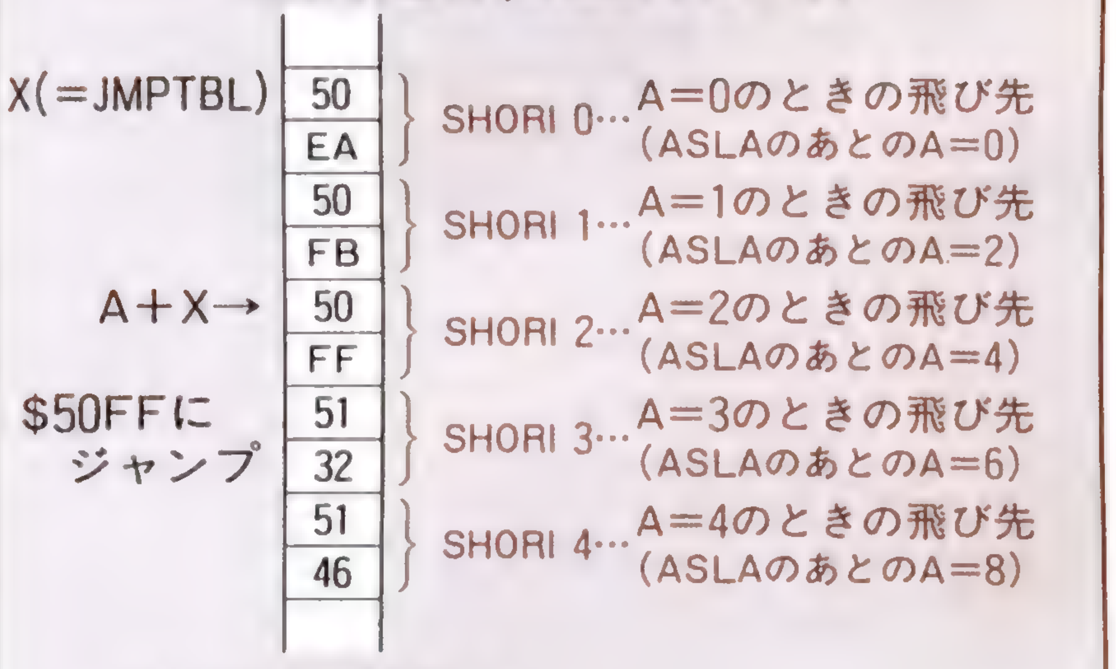
さて，このアドレッシングモードのメリットですが，

①LEA命令でインデックスレジスタ(X，Y，U，S)とアキュムレータ(A，B，D)の加算が行える

②テーブル参照が容易

などがあげられます。①は特に説明しなくてもよいでしょう。②のテーブル参照というのは，データの並び(テーブル)から目的のデータを取り出す作業のことをいいます。たとえば，$0～$FのASCIIコードを求めたいとき，"0"のASCIIコード，"1"のASCIIコード，…"F"のASCIIコードを順に並べておけば，数値0のASCIIコードは0番目，数値1のASCIIコードは1番目，……という具合に簡単に変換できます。このプログラムを**リスト5**に示しました。Aレジスタに0～15の値を入れてこのプログラムをサブルーチンコールすれば，Bレジスタにその数値に対応する文字("0"～"F")のASCIIコードが得られます。なおこのルーチン(プログラム)は単独では動かないので，試しに実行したい人は，何かテスト用のプログラムを作る必要があります。その例を**リスト6**に示します。

図23 A＝2で，このルーチンに入ったときの動作(ASLAでA＝4)



SHORI 0＝$50EA
1＝$50FB
2＝$50FF
3＝$5132
4＝$5146としたときの例

また，BASICのON～GOTOなどに相当する処理もテーブル参照で容易に行えます。今度はジャンプ先のアドレスを並べるのです。ただ，ジャンプ先アドレスは2バイト要するので，テーブル先頭アドレスから＋2，＋4とする必要があります。**リスト7**の例ではAレジスタに何番目のジャンプ先かという情報（0, 1, …）を入れておけば，そこへジャンプする例を示します（**図23**）。Aレジスタの値は0，1，2，…となっていることを前提としているので，ASLAでAの値を2倍している点に注意してください。

3 プログラムカウンタリラティブアドレッシング

条件分岐命令は，ジャンプ先を次の命令のあるアドレスからのオフセットで示しました（リラティブアドレッシング）。このアドレッシングモードと同じ方法で実効アドレスを求めるのがプログラムカウンタ（PC）リラティブです。

つまり，実効アドレスを，現在のPCの値との差（すなわちオフセット）で指定するのです。このオフセットは8ビット符号つきと16ビット符号つきの2とおりを選べます。式で表すと，

EA（実効アドレス）＝PC＋d　（d：8ビット，または16ビットの符号つき）

となります。インデックスレジスタとしてPCを指定できると考えてもよいでしょう。

なお，アセンブリ言語では，

LDA　d，PC……(a)

とは書かず，

LDA　実効アドレス，PCR……(b)

と記します。〝PCR〟はインデックスというよりも，〝PCリラティブ〟という意味で用いると考えてください。もちろんリラティブアドレッシングのときと同様，オフセットはアセンブラが計算してくれます。このときオフセットのサイズは自動的に8ビット，または16ビットとしてくれます。なお強制的に8ビット，または16ビットとしたいときは，

LDA　<実効アドレス，PCR…8ビット

LDA　>実効アドレス，PCR…16ビット

とします。

次に機械語コードですが，OPコードにインデックスアドレッシングのものを用い，次にPCリラティブを示すポストバイト（求めるのには7月号P130の表1を用います。表中×のついているビットは1，0どちらでもかまいません），そしてそのあとオフセットを1バイトまたは2バイト付けるのです。したがって8ビットオフセットなら3～4バイト，16ビットオフセットなら4～5バイトの命令となります（**図24**）。

オフセットの求め方ですが，リラティブの場合と同じく実効アドレス計算時にはもうPCの値は命令長分増えているので，ハンドアセンブルの場合，注意が必要です。オフセットの計算のしかたを含め，PCリラティブの動作を**図25**に示します。

このアドレッシングモードのメリットはプログラムをアドレスと無関係に位置できるということです。つまり必要に応じてプログラムを置くアドレスを変えることができるのです。こういったプログラムのことを「リロケータブルなプログラム」とか「ポジションインデペンデント（位置独立）なプログラム」と呼びます。

4 インダイレクトアドレッシング

アドレッシングモードのラストを飾るのはインダイレクトアドレッシングです。

インダイレクトアドレッシングとは，その名のとおり，実効アドレスが間接的に指定されます。実効アドレスはメモリ上に存在し，実効アドレスの入ったメモリのアドレスをアドレッシングモードで指定するのです（**図26**参照）。

この実効アドレスの入ったメモリのアドレスは，

図24　PCリラティブの命令長

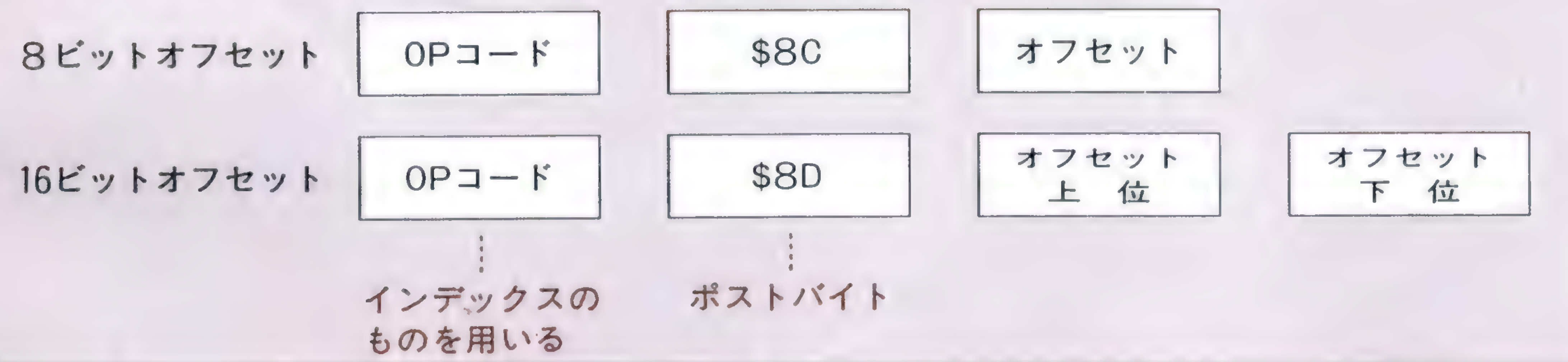


図25　PCリラティブの動作

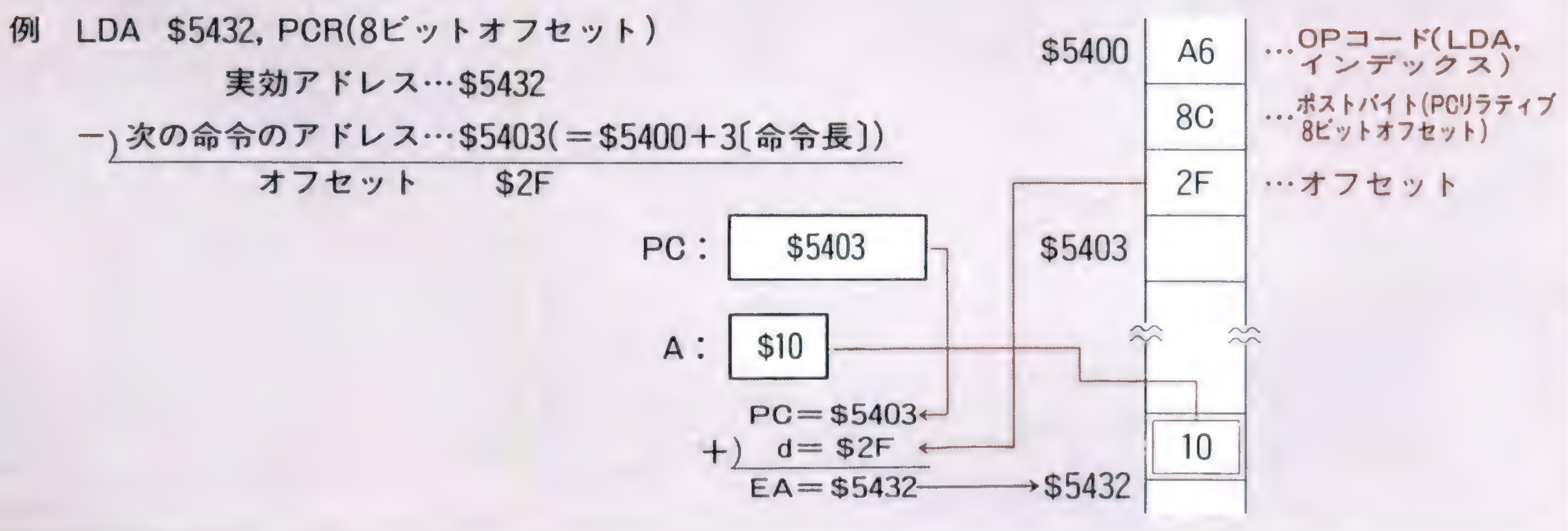
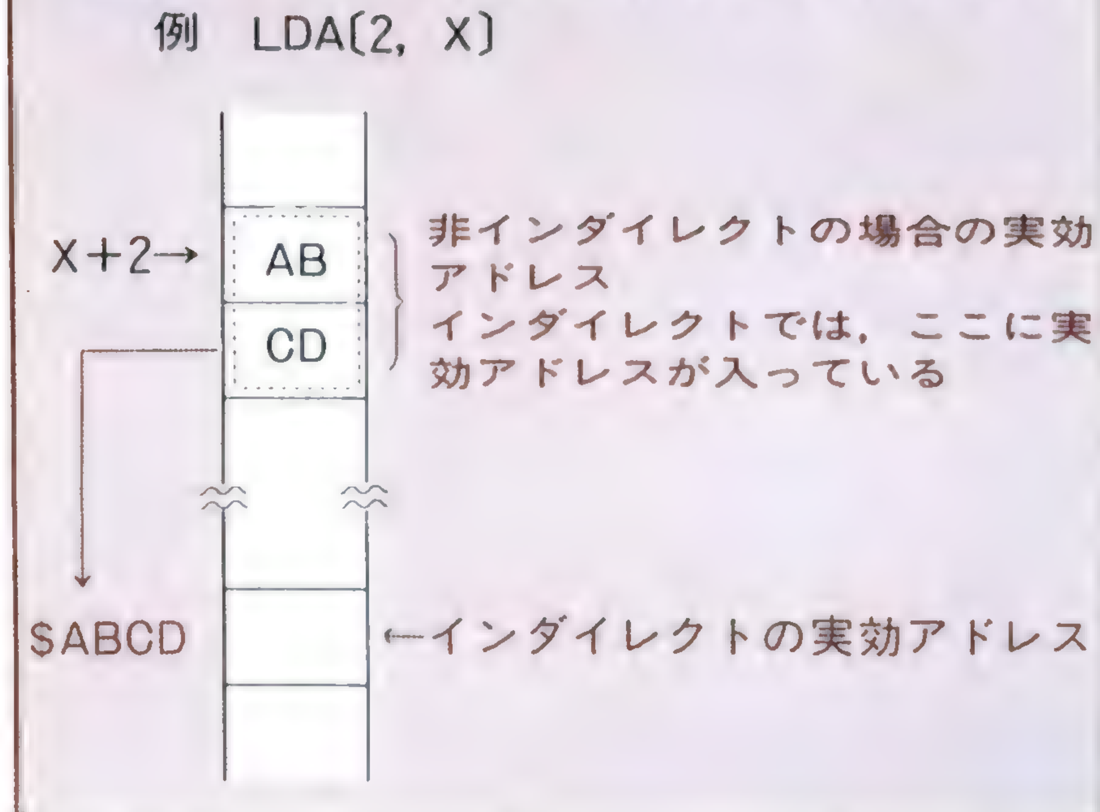


図26　インデックストインダイレクト



a. インデックス形式

- オフセットなし
- 8ビットオフセット
- 16ビットオフセット
- アキュムレータオフセット
- オフセットなし・ポストインクリメント(＋2),プリデクリメント(−2)付き

b. PCリラティブ形式

- 8ビットオフセット
- 16ビットオフセット

c. エクステンド形式

を用いることができます。もちろんこれらの方法で**実効アドレスの入ったメモリのアドレス**を指定するわけです。すなわちインデックスによってインダイレクトを行うという意味で**インデックストインダイレクトアドレッシング**(aやbのモード)，**エクステンデッドインダイレクトアドレッシング**(cのモード)と，これらを呼びます。動作例を図27に示します。

なお，オートインクリメント／デクリメントで＋1，−1がないのは，実効アドレスが2バイトで示されるからです。

これらのアドレッシングモードのアセンブリ言語での表記は，従来の(非インダイレクトの)オペランドを「〔」と「〕」でくくり，

(a) LDX 〔4, Y〕

(b) STA 〔PARAM1, PCR〕
└→ラベル

(c) JSR 〔$FBFA〕

のようにします。3番目はBIOSコールと同一，つまり，5章でお話ししたBIOSコールはJSRのエクステンデッドインダイレクトだったのです。$FBFAにはBIOSのエントリアドレスが入っていたのです。これは，BIOSのエントリが変わっても，$FBFAの内容を変更すればプログラムを変更する必要がないことを狙ったものです（実際FM-7，8，11でBIOSエントリアドレスは異なります)。

さて，次は機械語コード化の方法です。

まずOPコードはインデックスのものを使います（エクステンデッドインダイレクトの場合でも)。そして，細かなアドレッシングモードは，ポストバイト（これまた7月号P130表1参照）で表すのです。もちろんインダイレクトの方で見ます。なおこの表では見にくいので，**表3**にインダイレクトアドレッシングのポストバイト一覧を示しました。

ポストバイトの次は必要に応じて1バイト，または2バイトのオフセット（エクステンデッドインダイレクトのときは，2バイトのアドレス値)を付けます(図28参照)。

さて，このインダイレクトアドレッシングのメリットは，なんでしょうか。それはデータの入ったアドレスをデータとして処理できる，すなわちポインタの扱いが容易なことです。たとえば，高級言語Cのように変数の値，変数のアドレスといった処理が容易になります。複雑なアドレス計算で求まったアドレスの値をレジスタのかわりにメモリに入れても同じように処理できるのです。

また必要なデータのアドレスが連続していないとします。こういったとき，データのアドレスをメモリ上に並べれば，インダイレクトの使用であたかもデータが連続的に並んでいるかのように扱えるわけです。

データのアドレスをスタックに積んだときも，このアドレッシングモードは便利です。このように，インダイレクトアドレッシングは面倒ですが，使い方しだいでプログラムがエレガントになる可能性を持っています。ぜひマスタしてください。

◇　◇　◇

どうでしたか。今回の話は機械語でもわかりにくいところなので，じっくり考えて自分でいろいろ試してみてください。

次回は割り込みを中心に話を進める予定です。また，次回で6809の命令はすべて登場することになります。ご期待ください。

表3 インダイレクトアドレッシングのポストバイト

		インデックスレジスタ	X	Y	U	S	その他
インデックストインダイレクト	ゼロ オフセット		94	B4	D4	F4	—
	8ビットオフセット		98	B8	D8	F8	—
	16ビットオフセット		99	B9	D9	F9	—
	アキュームレータ・オフセット	[A, R]	96	B6	D6	F6	—
		[B, R]	95	B5	D5	F5	—
		[C, R]	9B	BB	DB	F13	—
	[, R＋＋]		91	B1	D1	F1	—
	[, −−R]		93	B3	D3	F3	—
プログラムカウンタリラティブ	8ビット		—	—	—	—	9C
	16ビット		—	—	—	—	9D
エクスランデットインダイレクト			—	—	—	—	9F

図27 インダイレクトの動作

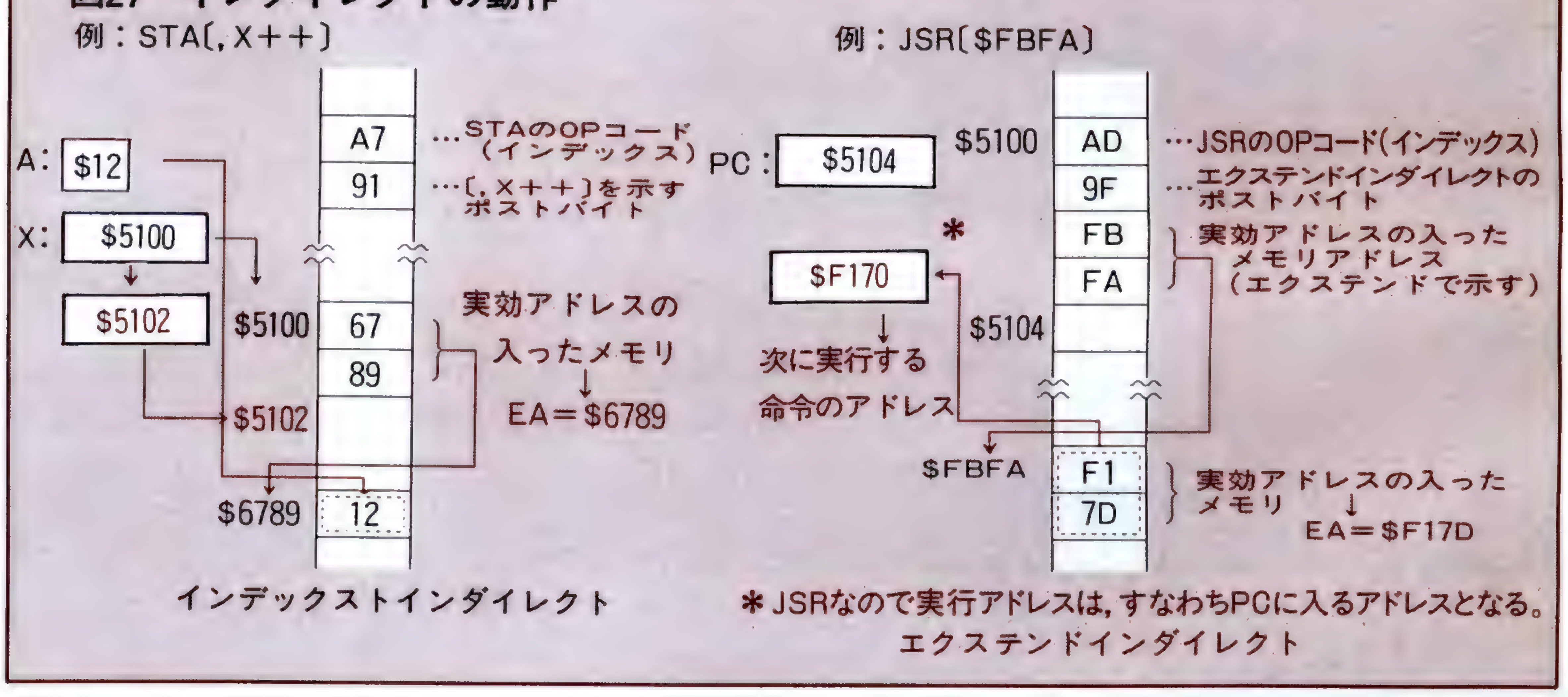
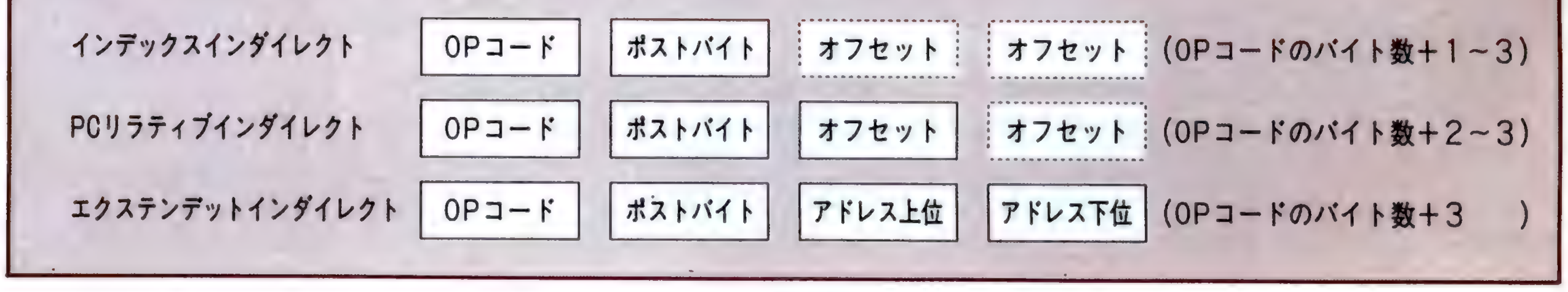


図28 インダイレクトの命令長



Oh!FM

はじめてさわるパソコン

小林　泰介

「習うより慣れろ」と申しますが，パソコンでも同じことです。カタコトだろうが，うろ覚えだろうが，とにかくいじくり回す。パソコンを埃をかぶった置物にしないための方法は，これが一番です。幸いなことに，パソコンの使い方を間違えても，火事になったり，骨折したりすることはありません。おまけに，『77をつかおう』，『F-BASIC入門』という親切な手引きがあるではありませんか。さあ！——というわけで，やっとNEW7や77を手に入れた人，新古・中古の7や8を発掘した人を対象にした，「はじめてさわるパソコン」を始めます。合言葉は「Syntax Error がなんだ，暴走がなんだ。僕らはパソコンを動かしたいんだ」です。

始める前に

最初にコードやケーブルの接続を確認してください。外れていたり，つなぎ方を間違えたりしていませんね。特にカセットテープレコーダ（データレコーダと呼びます）との接続はよく間違えるので注意してください。覚えるまでは図1のようにシールを張っておくといいでしょう。それから（つなぎ忘れたりして）新たに接続するときは，電源を切ってからつないでください。

確認が終わったら，電源スイッチを入れます（ONにします）が，これには順番があります。先にデータレコーダ，ディスプレイ，プリンタ，外部ディスクユニットなど（まとめて周辺機器と呼びます）の電源を入れて，最後にパソコン本体の電源を入れてください。パソコンを故障させないためですから習慣にしてください。

故障といえば，食べかすや飲みかすをこぼして壊すことがあります。パソコンを操作しながら飲み食いをするのはやめてください。また，タバコのヤニもパソコンに悪い影響を与えます。できたら操作しながらの喫煙もやめてください。

○設置場所の注意（マニュアル参照個所）	システム解説	取り扱い上のご注意　前文	
	77をつかおう	p14～p17	
○接続方法	システム解説	3－2～3－28	
	77をつかおう	p18～p32	

それでは電源を入れましょう。

たたく，打つ，打ち込む

ディスプレイの画面に，

```
FUJITSU F-BASIC Version 3.0 ——FM-8は1.0
Copyright (C) 1981 by FUJITSU/MICROSOFT

30530 Bytes Free

Ready
■
```

という文字が現われます。上の3行は，このBASICの名前とバージョン，版権表示，ユーザが使えるメモリの大きさを示しています。

下のReadyは「用意できました。なにか命令してください」というパソコンの合図です。命令は文字や記号の書いてあるボタン（ボタン一つひとつをキーと，ボタン全体をキーボードと呼びます）を押して伝えます。

伝えられた命令が画面に現れる位置を示すのは，点滅している白い四角（カーソルと呼びます）の仕事です。

ディスクを使うときは，3行の上に，

```
DISK VERSION
How many disk drives      ? 2⏎
How many disk files(0-15)? 2⏎
```

がつきます。上はディスクの数を尋ねているのでその数を（77D1の人やMB27607などシングルドライブの人は1，77D2の人やTF-10，MB27611などデュアルドライブの人は2）を押します。下はファイル（後日説明）の数を尋ねています。2ぐらいの数を押しておけばまずだいじょうぶです。また，ディスクを使うと使えるメモリの大きさが小さくなります。

```
25504 Bytes Free
```

では命令する前に指ならしをしましょう。適当にキーを押してみてください。

```
Ready
12345qwertASDFGﾂｻｿﾋｺ!"#$%&ｱｲｳｴｵﾔﾕﾖ+-*/?@
年月日時分秒¥円 ◣◤╳♠♥◆♣○●SAVE"SCREEN7,7
                            PF-キー  PE-キー
Syntax Error
Ready
```

いろいろな文字や記号が出たり，赤ランプがついたりしますね。

図1　オーディオカセットケーブルにシールを張ろう

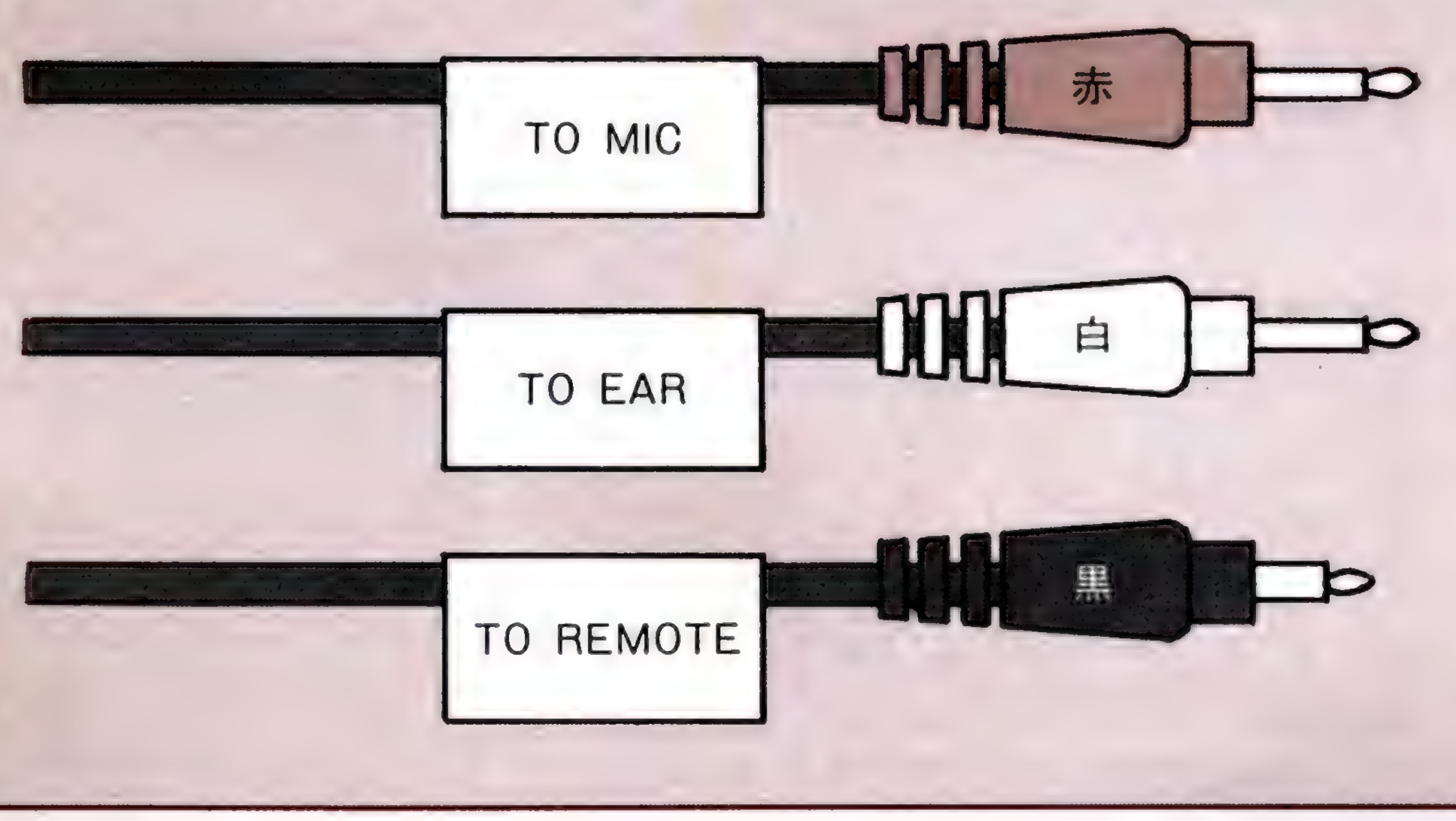
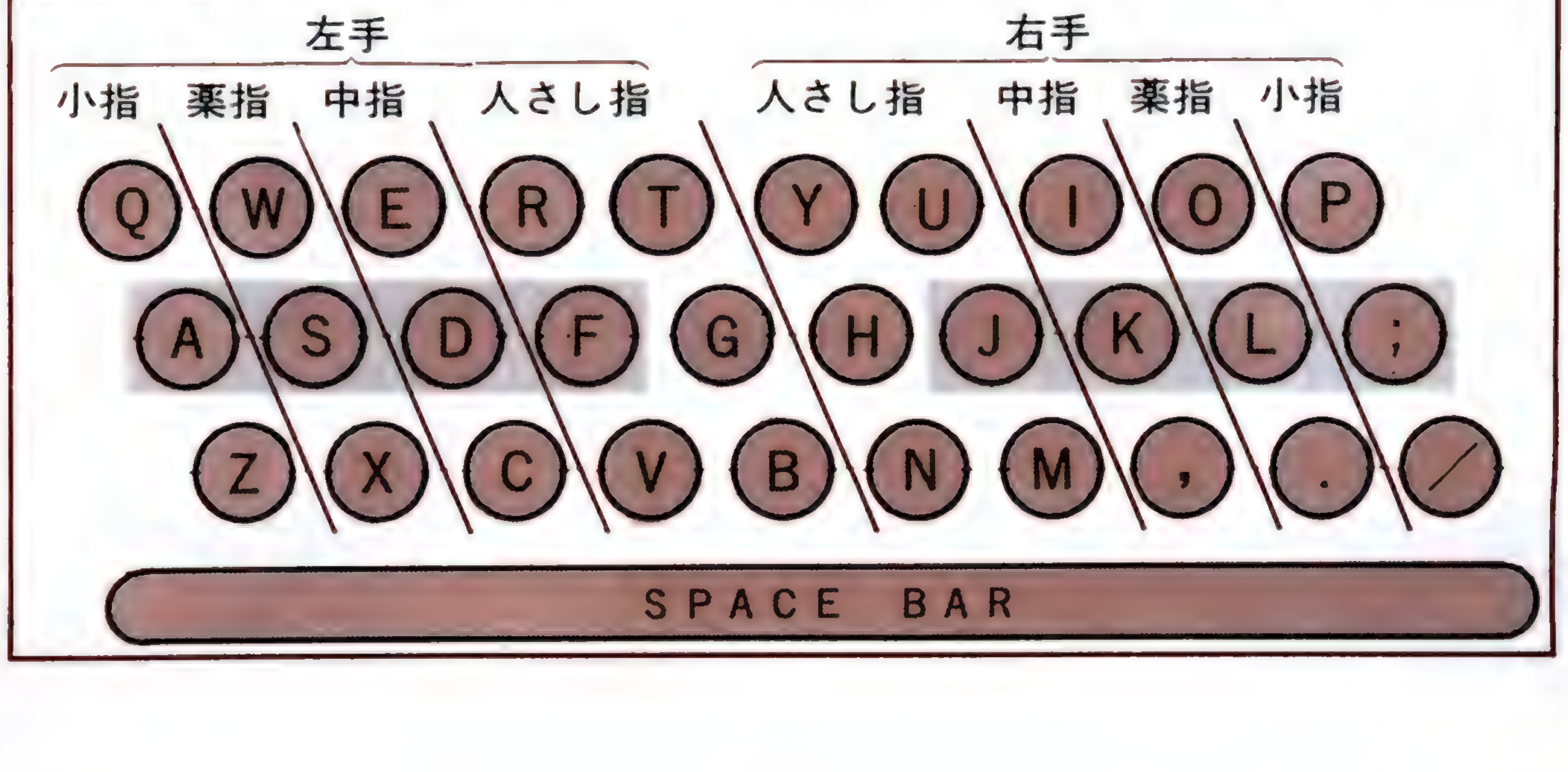


図2　指の使い分け（キーボードのASDFとJKL;はホームポジション）



gggggghhhjjjkkkkkkkl111111

と同じ文字が続けて出る人は，キーの押しすぎです。押したらすぐに指をはなしましょう。

キーはたたくような感じで押します。すばやく深く押して，すぐ指をはなします（このため，キーボードをたたく〈打つ，打ち込む〉という表現があります)。また，指の使いわけは図2のようにします（が，慣れるまでは人指し指1本でもかまいません)。

どのキーを打つとどの文字が出てくるかわかりましたか。パソコンのキー配列は，同じキーで3～6種類の文字や記号を表すことや，ABC順・50音順に並んでいないことなどのため，多少ややこしくなっています。慣れるしかないのですが，キーボードについて覚えてもらいたいことがあります。

ひとつは，パソコンのキーボードにはいくつかのモードがあり，モードを特殊なキーで切り換えて英大文字，英小文字，カナ，グラフィックを区別していることです。

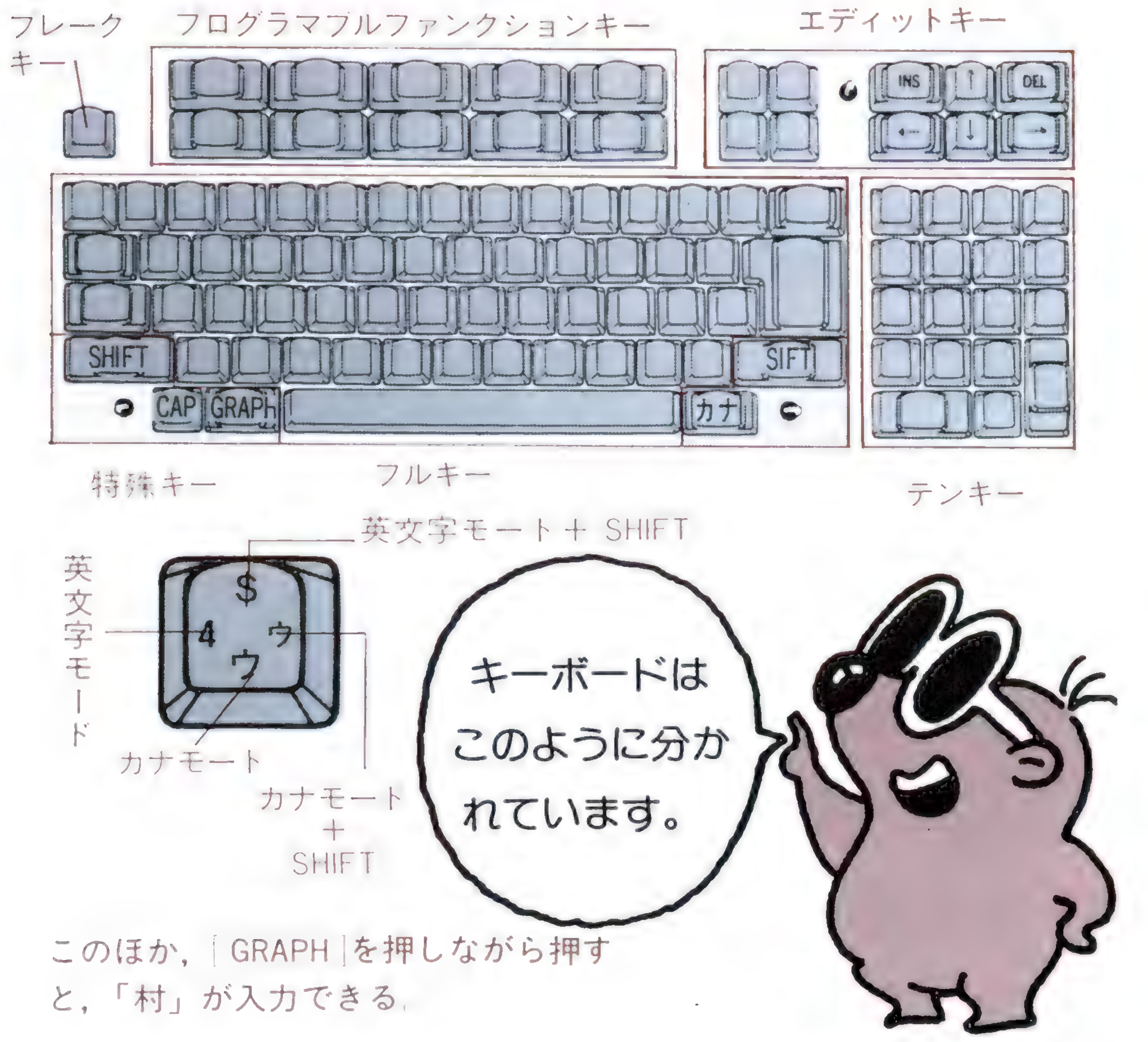
左下，赤ランプの隣りの［CAP］キーは，英大文字モードと英小文字モードを区別します。［CAP］キーを押し，隣の赤ランプ（CAPランプと呼ぶ）がついたあと，キーボードをたたくと画面に大文字が出ます。もう一度押してCAPランプが消えたあとは小文字が出ます。

同様に，右下の［カナ］キーを押すと，隣りの赤いカナランプがつき，カナモードになります。

！＂＃＄（）＜＞？などキーの上に書いてある記号を出したいときは，カナランプを消して両端にある［SHIFT］キーのどちらかを押し，押しつけたまま目的のキーを打ち，［SHIFT］キーをはなします。慣れてきたら同時に打ってください。

ァッャヲ。・を出したいときは，カナランプをつけて，同様に［SHIFT］キーを押したまま目的のキーを打ってください。

市区町村年月日時分秒円人の漢字12字と〒♠♥♦♣▁◢▐◥╪などのグラフィックパターンを出したいときは，［GRAPH］キーを押したまま目的のキーを打ってください。



このほか，［GRAPH］を押しながら押すと，「村」が入力できる。

○ディスクを使う	システム解説	3-22～3-26，4-3～4-5
	77をつかおう	p34～p37，p42，p48～p51
○キーボード	システム解説	3-3，5-1～5-18
	77をつかおう	p8～p9
	ハードウェア解説	p22～p25
○タイプ練習プログラム	FILES Oh!FM	p57(Vol.2 p33)
	7月号	p138

パソコンに命令を入れることを入力といい，入力に使うものを入力装置といいます。入力装置にはキーボードのほか，ジョイスティックなどがあります。逆にパソコンの処理した結果を出すことを出力といい，出力に使われるものを出力装置といいます。ディスプレイやプリンタがそうです。また，ディスクやデータレコーダのように入力，出力とも使えるものもあります。

パソコンに命令しよう

それではなにか命令を打ち込んでみましょう。その前にいろいろ打ち込んだデタラメを消去しましょう。77なら前面右，11なら右面前，ほかは背面右にあるRESETと書かれたスイッチを押してください。画面のデタラメが消えて，上の電源投入時と同じ状態になりましたね。この状態を初期状態といいます。また初期状態に戻すことを「リセットする」といいます。

では，［P］［R］［I］［N］［T］␣［5］↵と打ってください。␣はスペースバーを打ってください。画面上では空白（スペース）になります。末尾の↵はリターンキーを押してください。このリターンキーは命令を「実行せよ」という意味を持っています。

```
Ready
PRINT␣5↵
 5
```

PRINTとは，後にあるもの（数字，計算式の答，変数の代入値，文字，文字列など）を「画面に表示せよ」という命令です。上は数字の5を表示しろと命令して，5が現れたところです。

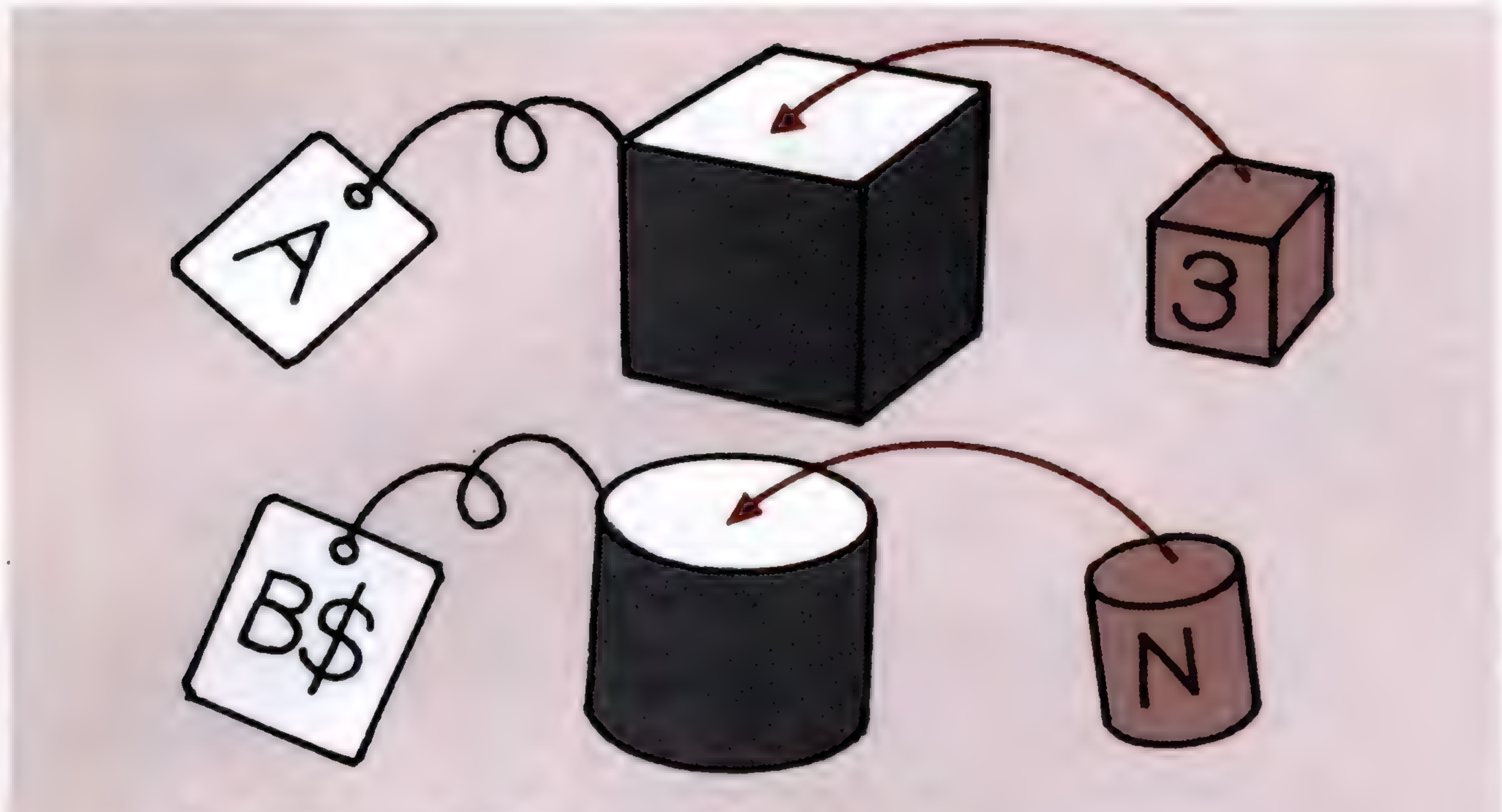
このPRINTは大文字でも小文字でも省略形の？でも同じ効果を示します。

```
Ready
print␣5↵
 5

Ready
?␣5↵
 5

Ready
```

数字のかわりに計算式をPRINTの後に置くと，計算式の答が表示されます。

```
Ready
PRINT␣3+5↵
 8

Ready
```

文字や文字の並び（文字列）を表示するときは，文字の前後に



数値変数と文字変数は型が違います。

「〃」(ダブルクォーテーション) をつけます。「〃」で挟まれたものはすべて文字として扱われます。たとえば計算式を「〃」で挟むと計算されないで，そのまま画面に表示されます。

```
Ready
PRINT "Naoko"⏎
Naoko

Ready
PRINT "3+5"⏎
3+5

Ready
```

では，LET A＝3⏎と打ったあとにPRINT A⏎と打ってみてください。

```
Ready
LET A=3⏎
Ready
PRINT A⏎
 3
Ready
```

LETは「右辺の数値・計算式の結果を，左辺の変数に代入せよ」という意味です。要は，AとかBという名前の箱 (左辺) に (右辺の) 数字をしまえ，ということです。画面は変化しませんが，ちゃんと変数Aが3になっています。PRINT A⏎と打ちAを表示させると，(Aの箱の中の) 3が表示されましたね。

ところで，3，5，8などの数値の左が1字ぶん空いていますね。これは正負の符号を入れるための空白です。

```
PRINT 3-5⏎
-2

Ready
```

○PRINT	F-BASIC 入門	p16～17
	F-BASIC 文法書	3-83～3-84

このように，そのつど命令を与え，そのつど実行する方式を直接モードといいます。それから，LETは省略可能です。

プログラムろう

パソコンには直接モードのほかに，一度にまとめて命令を与えてから，まとめて命令を実行する方式があり，間接モードと呼びます。例題を打ち込む前にCLS⏎を打ってください。瞬時に画面が消えた後，刹那(せつな)の差でReadyとカーソルが現れましたね。CLSは「画面をクリアせよ (画面上の文字や記号など消し去れ)」という命令です。パソコンの中味まで白紙にしてしまうリセットとは違いますので注意してください。

では，次のように打ってください。ただ，間接モードではリターンキーの意味が「今，打ち込んだものを覚えろ」と換わりますので，⏎を押してもその場で命令を実行しません。

```
10 CLS⏎
20 LET A=16⏎
30 LET B$="Naoko"⏎
40 PRINT B$;" is";A;" years old."⏎
50 END⏎
```

10，20，30…という番号が先頭についています。これは行番号というもので，①直接モードと間接モードを区別する，②命令を実行する順番を示す，の2つの役割があります。行番号には0から63999までの整数が使えますが，10か100か1000から10おきにするのが普通です。

30行のB$は文字を入れておく箱，文字変数です。数字を入れる箱 (数値変数) と文字を入れる箱 (文字変数) と区別するために，名前 (AとかBとか) の後に$をつけます。

40行の；(セミコロンと呼ぶ) は「区切る」記号です。PRINT命令の後では，文字や文字列や変数などを区切るとともに，それらを連続して (表示する行を変えたり，スペースを空けたりしないで) 表示することも示しています。

50行のENDは，「命令終わり！」という命令です。

それでは，一連のRUN⏎を打ち実行させましょう。

```
Naoko is 16  years old.

Ready
```

と画面に出ましたね。10行で画面を消したあと，20行でAという数値変数の箱に16を入れ，30行でB$という文字変数の箱にNaokoを入れ，40行でB$の中味のNaoko， is，Aの中味の16，years oldと続けて表示し，50行で終わりというプログラムです。

プログラムとはこのような一連の命令群のことです。

○CLS	F-BASIC 入門	p22
	F-BASIC 文法書	3-113
○LET	F-BASIC 入門	p18
	F-BASIC 文法書	3-60
○変数	F-BASIC 入門	p17，p28～29
	F-BASIC 文法書	2-5～2-6
○行	F-BASIC 入門	p15
	F-BASIC 文法書	2-1

後始末をしたあとは，改造しよう

このように画面に出なかった人は，どこかに打ち込みミスをしています。見付けて修正しましょう。

Readyの前に，

```
Syntax Error In 40
```

などと出てませんか。Syntax Error とは「プログラムの文の書

き方が間違っている」というパソコンのお小言（エラーメッセージと呼ぶ）です。Inのあとの数字が間違いのある行数です。

間違えた行数がわかったところで，LIST⏎と，「画面にプログラムを表示せよ」という命令を打ち込みます（リストを取るといいます）。LIST␣40とかLIST␣30-50と後に数字を付けるとその行だけが表示され，付けないと全行表示されます。

たとえば，

```
40 PRANT B$;" is";A;" years old."
```

と間違えていた場合は，右上にあるカーソル移動キー←↑↓→を使って（矢印の方向にカーソルが動きます），カーソルをAに重ね，Iと打ち直し，リターンキーを打ってください。AがIに直りましたね。

PRNTとIを抜かした場合は，インサートキー INS を押し，左の赤ランプがついたのを確認してから，カーソルをPRNTのNに重ねてIを打ち，リターンキーを押します。INSキーはなにかを挿入するときに使います。逆に削除するときは，削除する文字の直後にカーソルを動かし DEL キーを使います。

修正の後のリターンキーは，すべての間違いを直してから1回打つだけで充分です。

よく間違えるものにO（オー）と0（ゼロ），I（アイ）と1（イチ），B（ビー）と8（ハチ），C（シー）とG（ジー），，（カンマ）と．（ピリオド）；（セミコロン）と：（コロン），－（マイナス）とー（音引）があります。注意して打ち込んでください。

修正のあとは変更の仕方です。自分や友達の名前をB$に，年齢をAに入れるようにしましょう。

```
20␣INPUT␣"How␣old␣are␣you";A⏎
30␣INPUT␣"What␣is␣your␣name";B$⏎
```

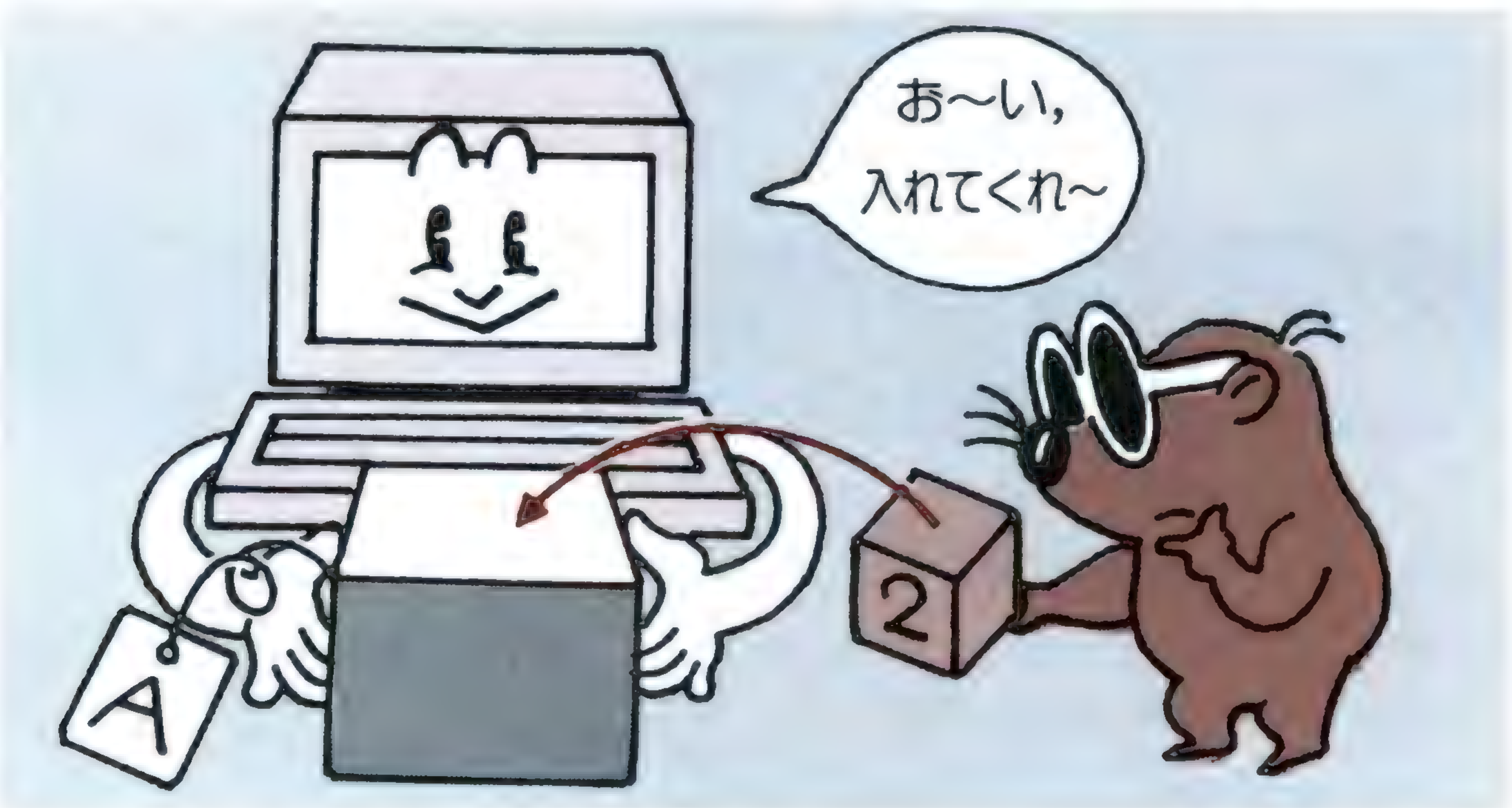
またLIST⏎と打ってください。20行と30行が置き換わりましたね。このように同じ行番号を新たに入力すると，古いほうが消されて新しいほうに置き換わります。

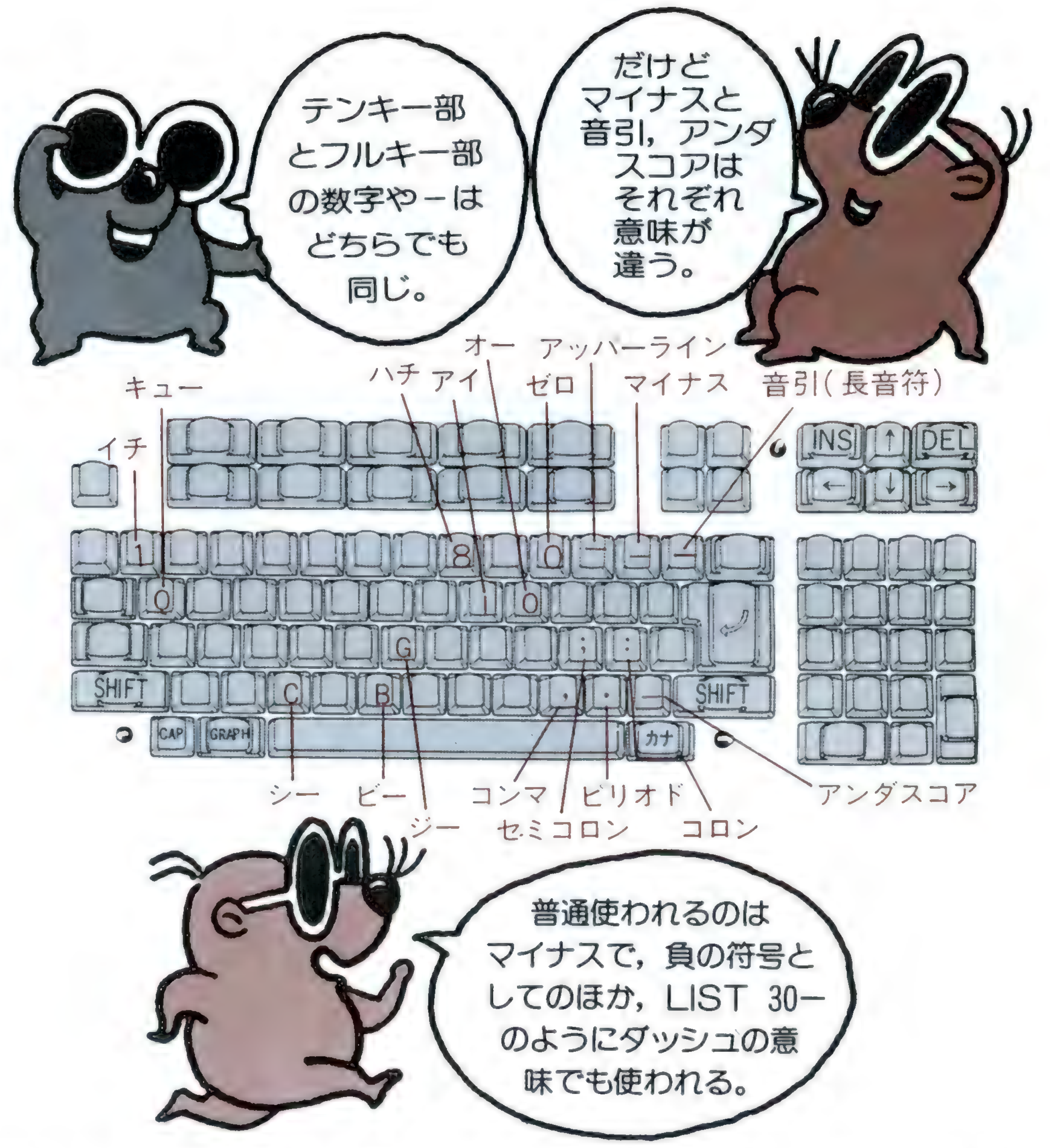
ではプログラムを実行してみましょう。年齢と名前を尋ねてきますので，キーボードをたたいて答えを入れてください。

```
Ready
How old are you? 18⏎
What is your name? T.KOBAYASHI⏎
T.KOBAYASHI is 18 years old.

Ready
```

となりましたね。

INPUTとは「キーボードから入力したデータを読み取れ」という命令です。INPUT命令に出会うと，パソコンは？を表示して，





あなたの指示を待つ状態になり，キーボードから必要な指示を与えてやると，それによって再び動作を再開します。INPUTの後の〝で挟まれた質問は，AなりB$なりの変数の箱になにを入れるのかを示して，わかりやすくするための文字列です。省略してもかまいませんが，できるだけ載せておくようにしましょう。質問を載せるときは，質問と変数名の間に「;」を必ず入れてください。

それから，数字を入れるところで文字を入れるなど変数の形式を間違えると，Redo from Startというエラーメッセージを出します。そのあとでもう一度同じ質問をしてきますので，今度は間違えないでください。

○エラーメッセージ	F-BASIC入門	p155
	F-BASIC文法書	付録-20
	システム解説	付録-18
○LIST	F-BASIC入門	p19
	F-BASIC文法書	3-6～3-7
○INPUT	F-BASIC入門	p30～p31
	F-BASIC文法書	3-80～3-81

おわりに

パソコンを使っていて「わからないな」と思ったら，即マニュアルをひもといてください。富士通のマニュアルはかなり親切に作ってあります。あなたがほしい情報のほとんどは載っているはずです。マニュアルを引く習慣をつけてください。

また，F-BASIC入門は良い教科書です。かたっぱしから例題のプログラムを打ち込み，間違えては直して覚えてください。

CP/M-86 BDOSコール入門(3)

西村　義孝

図A　ファイルコントロールブロック

dr	f1	f2	//	f8	t1	t2	t3	ex	s1	s2	rc	d0	//	dn	cr	r0	r1	r2
00	01	02	…	08	09	10	11	12	13	14	15	16	…	31	32	33	34	35

リスト1

©1984　Y. NISHIMURA

```
;
; BDOS CALL #23 ( Rename File )
;
        CSEG
        ORG     100H                         8080モデル

        MOV     AX,CS
        MOV     DS,AX

        MOV     CL,09H
        MOV     DX,OFFSET MESSAGE1           BDOS CALL……の表示
        INT     224

        CALL    PRINT_FILE_NAME              ファイル名を表示

        MOV     CL,23
        MOV     DX,05CH                      リネーム
        INT     224

        CMP     AL,0                         正常終了ならREN1へ
        JNE     REN1

        MOV     CL,09H
        MOV     DX,OFFSET MESSAGE2           ファイルがないことを通知
        INT     224
        JMP     ABORT
REN1:
        MOV     CL,09H
        MOV     DX,OFFSET MESSAGE3           "is renamed" を表示
        INT     224
ABORT:
        XOR     CL,CL
        XOR     DL,DL                        CCPへ
        INT     224
PRINT_FILE_NAME:                             FCBよりファイル名
        CLD             ! MOV   SI,05CH !LODSB     調べて表示
        CMP     AL,0    ! JE    PRINT_F1
        ADD     AL,'A'-1

        MOV     CL,2    ! MOV   DL,AL   ! INT   224
        MOV     CL,2    ! MOV   DL,':'  ! INT   224
PRINT_F1:
        MOV     CX,8
        CALL    PRINT1

        MOV     CL,2    ! MOV   DL,'.'  ! INT   224

        MOV     CX,3
        CALL    PRINT1
        RET
PRINT1:
        PUSH    CX
        LODSB           ! AND   AL,01111111B
        CMP     AL,' '  ! JE    PRINT2
        MOV     CL,2    ! MOV   DL,AL   ! INT   224
PRINT2:
        POP     CX
        LOOP    PRINT1
        RET

END_CS:
        DSEG
        ORG     OFFSET END_CS
MESSAGE1        DB      'BDOS CALL #23 ( Rename File )',0DH,0AH,'$'
MESSAGE2        DB      ' is renamed.','$'
MESSAGE3        DB      ' can not be found.','$'
        END
```

CP/M-86とMS-DOSはなにかにつけて比較されます。どちらがすぐれているかを決めることは難しいことですが，私自身はMS-DOSを使うことが多いようです。理由はやはりMS-DOSの方がディスクアクセスが速いためです。COPYをしてみればわかりますが，MS-DOSの方がはるかに速いはずです。これは，MS-DOSのBIOS部のブロックリード，ライト機能のためですが，COPYだけでなく，アセンブル，コンパイル，リンクなどディスクアクセスを含む処理は，すべて速くなっています。また，高性能なリンカが付属していますから，高級言語とアセンブリ言語のリンクが苦労なく行える点もMS-DOSを使う理由にあげられるでしょう。

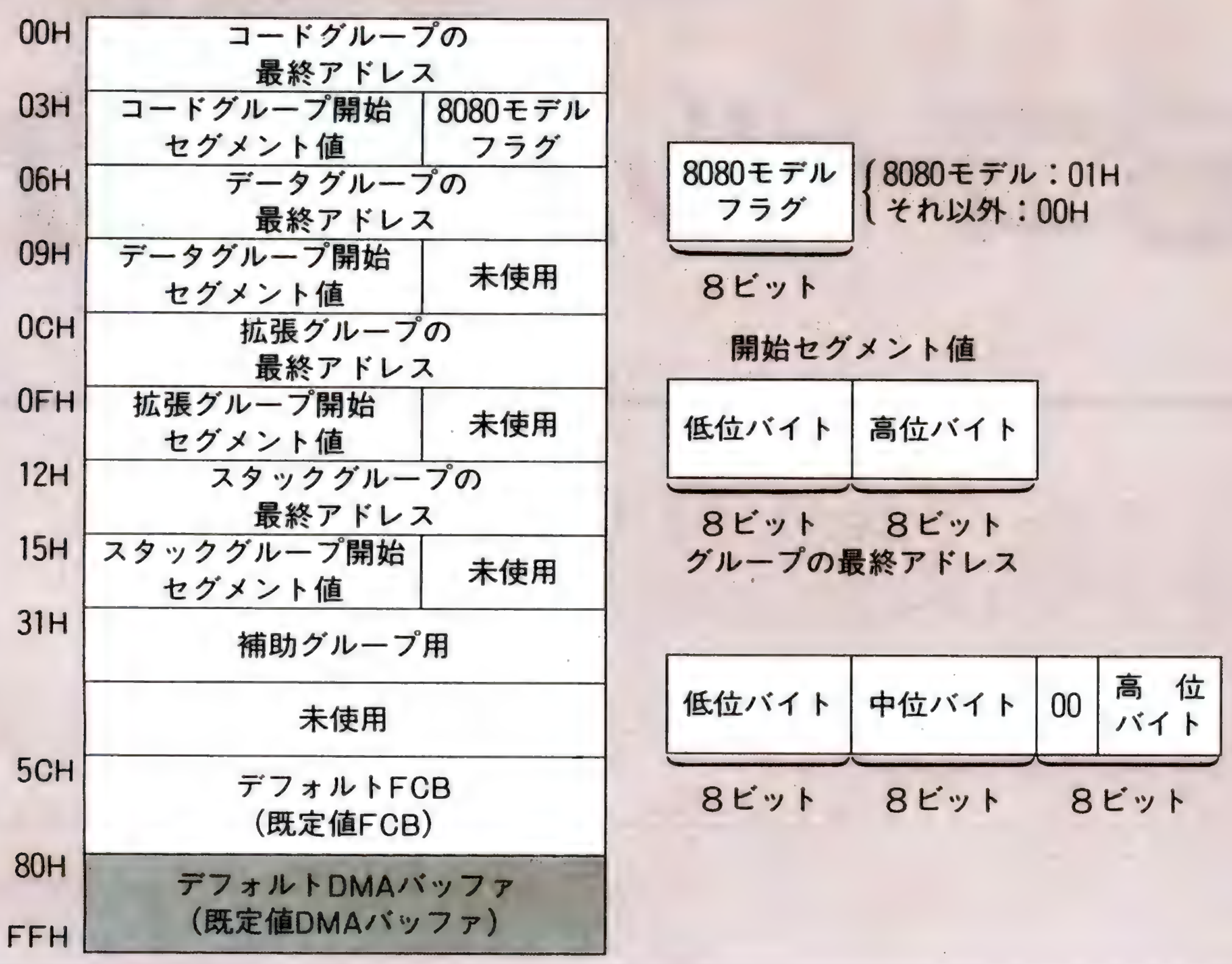
ではCP/M-86はまったく使わないかというと，そうでもないのです。私はアセンブルを行うときには，もっぱらCP/M-86を使っています。理由はSID86があるからです。また，ASM86は低機能ではありますが，アセンブルが高速な特徴もあります。

アセンブルとデバッグが高速に行えることは，やはり魅力の1つでしょう。私はCP/M-86上で完全にデバッグした後，MS-DOSのMASMにかけなおすというやり方を取っています。いかに高機能でも開発に

図1

```
A>
A>
A>dir temp?.dat
A: TEMP1    DAT
        TEMP1．DATがある

A>
A>ohfm23 temp1.dat temp2.dat
BDOS CALL #23 ( Rename File )
TEMP1.DAT is renamed.
        temp1.datをtemp2.datにリネーム

A>
A>dir temp?.dat
A: TEMP2    DAT
        確かにTEMP2．DATになっている

A>
A>ohfm23 temp1.dat temp2.dat
BDOS CALL #23 ( Rename File )
TEMP1.DAT can not be found.
A>
A>
        TEMP1．DATはない
```

図B ベースページの内容



時間がかかるのは考えものです（それは，White Smith社のＣコンパイラを見ているとよくわかります)。

CP/M-86とMS-DOSは，使いわけるのが一番ではないでしょうか。またCP/M-86とMS-DOSのBDOS コール(MS-DOSではファンクションコールと呼ばれています）は非常に似ていますから，CP/M-86のBDOS コールを勉強することは，かなり有意義ではないかと思います。

リスト2

```
;
; BDOS CALL #24 ( Return Login Vector )
;
        CSEG                                  8080モデル
        ORG     100H

        MOV     AX,CS
        MOV     DS,AX

        MOV     CL,09H                        BDOS CALL……の表示
        MOV     DX,OFFSET MESSAGE1
        INT     224

        MOV     CX,09H                        Login Vector：の表示
        MOV     DX,OFFSET MESSAGE2
        INT     224

        MOV     CL,18H                        Login Vectorを得る
        INT     224

        CALL    PRINT_BX_IN_BINARY            BXレジスタを２進表示

        XOR     CL,CL                         CCPへ
        XOR     DL,DL
        INT     224

PRINT_BX_IN_BINARY:                           BXレジスタを２進表示するルーチン
        MOV     CX,16
PRINT1:
        SHL     BX,1
        JB      PRINT2

        MOV     DL,'0'
        JMPS    PRINT3
PRINT2:
        MOV     DL,'1'
PRINT3:
        PUSH    BX
        PUSH    CX

        MOV     CL,02
        INT     224

        POP     CX
        POP     BX
        LOOP    PRINT1

        MOV     CL,02
        MOV     DL,'B'
        INT     224


        RET
END_CS:
        DSEG
        ORG     OFFSET END_CS
MESSAGE1        DB      'BDOS CALL #24 ( Return Login Vector )',0DH,0AH,
'$'
MESSAGE2        DB      'Login Vector :','$'
        END
```

23 ファイル名の変更

Rename File

使い方

MOV　CL，17H

MOV　DX，FCBのオフセット

INT　224

ALレジスタにリターンコード

AL＝0　…正常終了

AL＝255　…ファイルがない

BDOSコール23の「ファイル名の変更」は，FCBのdr～t3フィールドで示されるファイルのファイル名をd_0～d_{15} フィールドで示されるファイル名に変更するものです。CP/M-86のコマンドの RENはこの機能によって簡単に実現できます。

リターンコードは，dr～t3フィールドのファイルが存在する場合０が入り，存在しない場合は255となります。

CP/M-86 ではコマンドに続いてファイル名を２つ置くとそれらは自動的にデフォルトFCBにセットされます。たとえば，

COPY TEMP.DAT TEST.DAT⏎

とすると，COPYというコマンドが呼ばれた時点で，デフォルトFCBのdr～t3 フィールドに TEMP.DATがセットされ，d_0～d_{15} フィールドに TEST.DAT というファイル名が自動的にセットされます。ですから REN に相当するコマンドは非常に簡単に作れます。

リスト1にRENと同等のプログラムを，その実行例を**図1**に載せておきます。

FCBの構造をまた**図A**に示します。思い出してください。

これまでデフォルトFCBを説明せずに使ってきましたが，これはどこにあるのでしょうか。CP/M-86のユーザプログラムの先頭の256バイトは

「ベースページ」

と呼ばれ，ユーザプログラムのメモリ配置状況，デフォルトDMAバッファなどCCPとユーザプログラムとのインタフェースを行う場所になっています。実はデフォルトFCBもこのベースページに含まれているのです。**図B**にベースページの内容を示します。この図からもわかるように，デフォルトFCBは，ユーザプログラムの先頭から5CHのところより始まっています。このことは知らない人も多いようなので覚えておいてください。CP/M-86でアセンブルを行うときに

ORG 100H

とする理由もそこにあります。もしORG 0Hとすれば，ベースページがユーザプログラムの先頭256バイトを破壊してしまいます（CP/M-86上で実行しないプログラムならもちろんORG 100Hにこだわる必要はありません）。

24 ログインベクトルの取り出し

Return Login Vector

使い方

MOV CL, 18H

INT 224

BXにログインベクトルが入る

ログインベクトルというのは，どのドライブが**オンライン**になっているかを示す情報です。オンラインとは，簡単にいえばアクセスされたことを意味します。オンラインとなっているドライブはディレクトリ情報が，CP/M-86側で把握されています。

図2

```
A>
A>

A>stat

A: RW, Free Space:        42k

A>ohfm24
BDOS CALL #24 ( Return Login Vector )
Login Vector :0000000000000001B
A>                              ← Aドライブのみオンラインとなっている
A>b:
B>
B>a:stat

A: RW, Free Space:        42k    } A, Bがオンラインのときは……
B: RW, Free Space:        80k

B>a:ohfm24
BDOS CALL #24 ( Return Login Vector )
Login Vector :0000000000000011B
B>
B>                              ← 確かにA, Bがオンラインであることがわかる
```

リスト3

```
;
; BDOS CALL #25 ( Return Current Disk )
;
        CSEG
        ORG     100H            } 8080モデル

        MOV     AX,CS
        MOV     DS,AX

        MOV     CL,09H
        MOV     DX,OFFSET MESSAGE1      } BDOS……の表示
        INT     224

        MOV     CX,09H
        MOV     DX,OFFSET MESSAGE2      } "Current Disk="の表示
        INT     224

        MOV     CL,19H          } カレントディスクの番号を得る
        INT     224

        ADD     AL,'A'
        MOV     DL,AL           } ドライブ名を表示
        MOV     CL,2
        INT     224

        XOR     CL,CL
        XOR     DL,DL           } CCPへ
        INT     224

END_CS:
        DSEG
        ORG     OFFSET END_CS
MESSAGE1        DB      'BDOS CALL #25 ( Return Current Disk )',0DH,0AH,
'$'
MESSAGE2        DB      'Current Disk =','$'
        END
```

図3

```
A>
A>ohfm25
BDOS CALL #25 ( Return Current Disk )
Current Disk =A
A>              Aドライブ
A>b:Bを選ぶ
B>a:ohfm25
BDOS CALL #25 ( Return Current Disk )
Current Disk =B
B>              確かにBドライブ
B>
```

リスト4

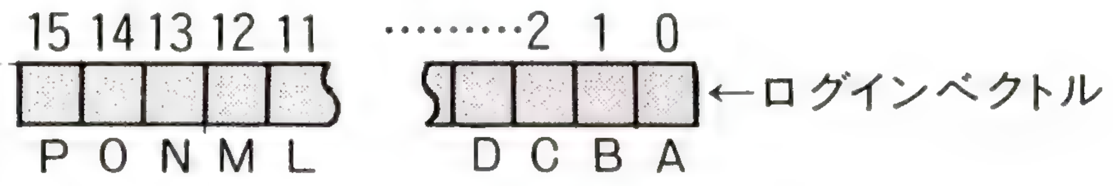
```
;
; BDOS CALL #26 ( Set DMA Address )
;
        CSEG
        ORG     100H

        MOV     AX,CS   ! MOV   DS,AX

        MOV     CL,09H
        MOV     DX,OFFSET MESSAGE1
        INT     224

        MOV     CL,15
        MOV     DX,05CH
        INT     224

        CMP     AL,255
        JE      FILE_DOES_NOT_EXIST
        MOV     BYTE PTR .5CH+32,0

        MOV     CL,1AH
        MOV     DX,OFFSET DMA_BUFFER
        INT     224

        MOV     CL,20   ! MOV   DX,5CH  ! INT   224
        CALL    PRINT_128_CHARACTOR

        XOR     CL,CL
        XOR     DL,DL
        INT     224

PRINT_128_CHARACTOR:
        CLD
        MOV     SI,OFFSET DMA_BUFFER
        MOV     CX,128
PRINT1:
        PUSH    CX
        LODSB
        CMP     AL,1AH  ! JNE   PRINT2
        POP     CX      ! RET
PRINT2:
        MOV     CL,2    ! MOV   DL,AL   ! INT   224
        POP     CX
        LOOP    PRINT1
        RET

FILE_DOES_NOT_EXIST:
        MOV     CL,09H
        MOV     DX,OFFSET MESSAGE2
        INT     224
ABORT:
        XOR     CL,CL   ! XOR   DL,DL   ! INT   224
END_CS:
        DSEG
        ORG     OFFSET END_CS
DMA_BUFFER      RB      128

MESSAGE1        DB      'BDOS CALL #26 ( Set DMA Address )',0DH,0AH,'$'
MESSAGE2        DB      ' File can not be opened.','$'
        END
```

- 8080モデル（4～7行）
- "BDOS CALL……"の表示（9～11行）
- オープンファイル（13～15行）
- AL＝255ならばファイルがない（17～18行）
- CRフィールドを0に（19行）
- DMAバッファを変更(DMA_BUFFERに)（21～23行）
- シーケンシャルリード（25行）
- 128文字を表示（26行）
- CCPへ（28～30行）
- DMA_BUFFERからの128文字を表示するチーチン（32～45行）
- CCPへ（52行）

図4

```
A>ohfm26 ohfm26.a86
BDOS CALL #26 ( Set DMA Address )
;
; BDOS CALL #26 ( Set DMA Address )
;
        CSEG
        ORG     100H

        MOV     AX,CS   ! MOV   DS,AX

        MOV     CL,09H
        MOV     DX,OFFSET MESSAGE1
A>
A>ohfm26 ohfm26.asm
BDOS CALL #26 ( Set DMA Address )
 File can not be opened.
A>
```

ログインベクトルは，最下位ビットがドライブAに，最上位ビットがドライブPに相当します。つまり



となります。各ビットが

0……オンラインでない

1……オンライン

となっています。

リスト2に「ログインベクトルの取り出し」の例を示します（実行例**図2**）。

25 選択されているディスク番号の取り出し

Return Current Disk

使い方

MOV　CL, 19H

INT　224

AL にドライブ番号

AL……0＝A　1＝B　……　15＝P

BDOSコール25番は，現在選択されているディスクの番号を返します。このディスクをカレントディスクと呼びます。前項のオンラインのディスクは複数存在しますが，カレントディスクはただ1つです。**リスト3**に例を示します（実行例**図3**）。

26 DMAアドレスのセット

Set DMA Address

使い方

MOV　CL, 1AH

MOV　DX, DMA のオフセット

INT　224

CP/M-86でのDMAとはDirect Memory Addressの意味で，ディスクとのやりとりを行う入出力バッファのことです。CP/M-86では1レコードが128バイトですから，DMA バッファも128バイトとなります。ベースページのところでも述べたように，デフォルト状態ではDMA バッファはベー

スページの先頭から80Hのところに決まっています。この位置をほかのところに移したいときに「DMA アドレスのセット」を行います。一度変更されると，リブートされるか，新たに変更しない限りDMAアドレスは変化しません。

BDOSコール26番の例を**リスト4**に示します（実行例**図4**）。

8086ではメモリの位置はセグメントとオフセットで表されますが，BDOSコールの26番ではオフセットしか指定できません。セグメントの変更はBDOSコール51番（Set DMA BASE）で行います。

27 アロケーションアドレスの取り出し

Get Addr (Alloc)

使い方

MOV　CL，1BH
INT　224
BX：　アロケーションオフセット
ES：　セグメントベース

アロケーションアドレスとは，アロケーションベクトルの位置です。アロケーションベクトルとは，ディスク上のどの部分が使われていて，どこが未使用かを示します。ですから，この機能でディスクの残りの容量を調べることができます。ディスクの諸元によってアロケーションベクトルの大きさは変わってきます。

1データブロック（ファイルの最少単位で1Kバイトや2Kバイト程度）のファイルがセーブされると，アロケーションベクトルのあるバイトは，

1000　0000　＝80H

となります。以下同様に，1データブロックセーブされるごとに，

1100　0000　＝　C0H
1110　0000　＝　E0H
1111　0000　＝　F0H
⋮
1111　1111　＝　FFH

リスト5

```
;
; BDOS CALL #27 ( Get Addr (Alloc))
;
        CSEG                                     } 8080モデル
        ORG     100H
        
        MOV     AX,CS   ! MOV   DS,AX
        
        MOV     CL,09H                           } "BDOS CALL…"の表示
        MOV     DX,OFFSET MESSAGE1
        INT     224
        
        MOV     CL,09H                           } "ALOCATION VECTOR:"の表示
        MOV     DX,OFFSET MESSAGE2
        INT     224
        
        MOV     CL,1BH                           } ES, BXにアロケーションアドレスを得る
        INT     224
        
        CALL    PRINT_ALLOC                      アロケーションベクトルの表示
        
        XOR     CL,CL                            } CCPへ
        XOR     DL,DL
        INT     224
        
PRINT_ALLOC:                                     } アロケーションベクトルの
        MOV     CX,10                              先頭10バイトを16進で表示
PRINT_1:
        PUSH    CX
        
        MOV     AL,ES:[BX]      ! INC BX
        CALL    PRINT_AL_IN_HEX
        
        POP     CX
        LOOP    PRINT_1
        
        RET
        
PRINT_AL_IN_HEX:                                 } ALレジスタの値を16進で表示するルーチン
        PUSH    ES
        PUSH    BX
        
        MOV     AH,AL
        AND     AL,0F0H
        MOV     CL,4
        SHR     AL,CL
        CALL    PRINT_AL0
        
        MOV     AL,AH
        AND     AL,0FH
        CALL    PRINT_AL0
        
        MOV     DL,' '
        MOV     CL,2
        INT     224
        
        POP     BX
        POP     ES
        RET
PRINT_AL0:
        CMP     AL,10   ! JAE   PRINT_AL1
        ADD     AL,'0'
        JMPS    PRINT_AL2
PRINT_AL1:
        ADD     AL,'A'-10
PRINT_AL2:
        PUSH    AX
        MOV     DL,AL
        MOV     CL,2
        INT     224
        POP     AX
        RET
        
END_CS:
        DSEG
        ORG     OFFSET END_CS
MESSAGE1        DB      'BDOS CALL #27 ( Get Addr(Alloc) )',0DH,0AH,'$'
MESSAGE2        DB      'ALLOCATION VECTOR : ','$'
        END
```

図5

```
A>
A>OHFM27
BDOS CALL #27 ( Get Addr(Alloc) )
ALLOCATION VECTOR : FF FF FF FF FF FF FF FF FF FF ←多くのファイルがある場合
A>
A>B:
B>A:OHFM27
BDOS CALL #27 ( Get Addr(Alloc) )
ALLOCATION VECTOR : FC 00 00 00 00 00 00 00 00 00 ←ファイルが少ない場合
B>
B>
```

リスト6

```
;
; BDOS CALL #28 ( Write Protect Disk )
;
        CSEG                                  } 8080モデル
        ORG     100H
        
        MOV     AX,CS   ! MOV   DS,AX

        MOV     CL,09H                        } "BDOS CALL…"の表示
        MOV     DX,OFFSET MESSAGE1
        INT     224

        MOV     CL,1CH                        } ライトプロテクト
        INT     224

        MOV     CL,09H                        } "Current Disk…"
        MOV     DX,OFFSET MESSAGE2
        INT     224

        XOR     CL,CL                         } CCPへ
        XOR     DL,DL
        INT     224

END_CS:
        DSEG
        ORG     OFFSET END_CS
MESSAGE1        DB      'BDOS CALL #28 ( Write Protect Disk )',0DH,0AH,'$'
MESSAGE2        DB      'Current Disk is write-protected. ','$'
        END
```

リスト7

```
;
; BDOS CALL #29 ( Get Read/Only Vector )
;
        CSEG                                  } 8080モデル
        ORG     100H

        MOV     AX,CS
        MOV     DS,AX

        MOV     CL,09H                        } "BDOS…"の表示
        MOV     DX,OFFSET MESSAGE1
        INT     224

        MOV     CX,09H                        } "Read/Only Vector :"の表示
        MOV     DX,OFFSET MESSAGE2
        INT     224

        MOV     CL,1DH                        } BXにR/Oベクトルを得る
        INT     224

        CALL    PRINT_BX_IN_BINARY            BXを2進表示

        XOR     CL,CL                         } CCPへ
        XOR     DL,DL
        INT     224

PRINT_BX_IN_BINARY:                           } BXレジスタを2進で表示
        MOV     CX,16
PRINT1:
        SHL     BX,1
        JB      PRINT2

        MOV     DL,'0'
        JMPS    PRINT3
PRINT2:
        MOV     DL,'1'
PRINT3:
        PUSH    BX
        PUSH    CX

        MOV     CL,02
        INT     224

        POP     CX
        POP     BX
        LOOP    PRINT1

        MOV     CL,02
        MOV     DL,'B'
        INT     224


        RET
END_CS:
        DSEG
        ORG     OFFSET END_CS
MESSAGE1        DB      'BDOS CALL #29 ( Get Read/Only Vector )',0DH,0AH,'$'
MESSAGE2        DB      'Read/Only Vector :','$'
        END
```

というふうに進み，FFHとなったあとは，次のバイトが80Hとなり，またC0H,E0H F0H……FFHと進んでいきます。

1ビットが何Kバイトに相当するかは，ディスクパラメータテーブル内のbls(＝1データブロックのサイズ）によって決まります。**リスト5**に，アロケーションベクトルの先頭 10 バイトを表示する例を示します（実行例**図5**)。

28 ディスクのライトプロテクト

Write Protect Disk

使い方

MOV　CL, 1CH
INT　224

これは，カレントディスクを一時的に書き込み保護します。CTRL-Cやディスクのリセットなどを行うと，これは解除されます。

無理に書き込もうとすれば例の

BDOS ERR ON d：R/O

と表示されます。

リスト6に，BDOSコールの28番の例を示します（実行例**図6**)。

図6

```
A>STAT

A: RW, Free Space:          40k
   └──────────────────→ Aドライブは R/W

A>OHFM28
BDOS CALL #28 ( Write Protect Disk )
Current Disk is write-protected.
                          ライトプロテクト化
A>
A>STAT

A: RO, Free Space:          40k
   └──────────────────→ 確かにR/Oになっている

A>
A>
```

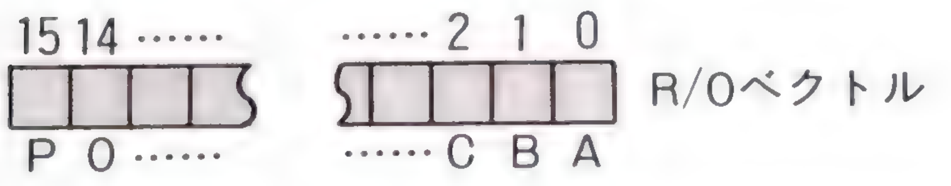
29 リードオンリーベクトルの取り出し

Get Read/Only Vector

使い方

MOV CL, 1DH
INT 224
BX に R/O ベクトルが入る

「リードオンリーベクトルの取り出し」はリードオンリーとなっているドライブを示す R/O ベクトルを取得します。R/Oベクトルの構成は，ログインベクトルと同様



となっています。各ビットの意味は，

0 ……R/W
1 ……R/O

です。**リスト7**にR/Oベクトルの表示の例を示します（実行例図7）。

リスト8

```
;
; BDOS CALL #30 ( Set File Attributes )
;
        CSEG                              } 8080モデル
        ORG     100H

        MOV     AX,CS
        MOV     DS,AX

        MOV     CL,09H                    } "BDOS…"の表示
        MOV     DX,OFFSET MESSAGE1
        INT     224

        CALL    PRINT_FILE_NAME           ファイル名表示

        OR      BYTE PTR .5CH+09,10000000B     t'フィールドを1に

        MOV     CL,1EH                    } アトリビュートをセット(R/Oに)
        MOV     DX,05CH
        INT     224

        CMP     AL,0                      } AL≠0ならファイルがない
        JNE     SET1

        MOV     CL,09H                    } "is R/O…"を表示
        MOV     DX,OFFSET MESSAGE2
        INT     224
        JMP     ABORT
SET1:
        MOV     CL,09H                    } "can not…"を表示
        MOV     DX,OFFSET MESSAGE3
        INT     224
ABORT:
        XOR     CL,CL                     } CCPへ
        XOR     DL,DL
        INT     224
PRINT_FILE_NAME:                          } FCBよりファイル名を得て表示する
        CLD             ! MOV   SI,05CH !LODSB
        CMP     AL,0    ! JE    PRINT_F1
        ADD     AL,'A'-1

        MOV     CL,2    ! MOV   DL,AL   ! INT   224
        MOV     CL,2    ! MOV   DL,':'  ! INT   224
PRINT_F1:
        MOV     CX,8
        CALL    PRINT1

        MOV     CL,2    ! MOV   DL,'.'  ! INT   224

        MOV     CX,3
        CALL    PRINT1
        RET
PRINT1:
        PUSH    CX
        LODSB           ! AND   AL,01111111B
        CMP     AL,' '  ! JE    PRINT2
        MOV     CL,2    ! MOV   DL,AL   ! INT   224
PRINT2:
        POP     CX
        LOOP    PRINT1
        RET

END_CS:
        DSEG
        ORG     OFFSET END_CS
MESSAGE1        DB      'BDOS CALL #30 ( Set File Attributes )',0DH,0AH,
'$'
MESSAGE2        DB      ' is R/O.','$'
MESSAGE3        DB      ' can not be found.','$'
        END
```

図7

```
A>
A>STAT

A: RW, Free Space:        40k
B: RW, Free Space:       976k
                        A,BともにR/Wのときは…
A>OHFM29
BDOS CALL #29 ( Get Read/Only Vector )
Read/Only Vector :0000000000000000B
                        確かにR/W ───↑

A>
A>STAT B:=[R/O]

Drive B: set to Read Only (RO)
                        ところがBをR/Oにすると

A>
A>OHFM29
BDOS CALL #29 ( Get Read/Only Vector )
Read/Only Vector :0000000000000010B
                   確かにBがR/Oになっている ─↑
A>
A>
```

図8

```
A>
A>STAT TEMP

 Drive A:                            User :  0
 Recs   Bytes FCBs Attributes        Name
    1     2k     1 Dir RW          A:TEMP    .
-----------------------------------------------
                        └──────────── TEMPはR/W
Total:     2k   .  1
A: RW, Free Space:          38k
A>
A>OHFM30 TEMP                          } R/Oのアトリビュートをつける
BDOS CALL #30 ( Set File Attributes )
TEMP. is R/O.
A>
A>STAT TEMP

 Drive A:                            User :  0
 Recs   Bytes FCBs Attributes        Name
    1     2k     1 Dir RO          A:TEMP    .
-----------------------------------------------
                        └──────────── 確かにTEMPはR/Oとなった
Total:     2k      1
A: RW, Free Space:          38k
A>
A>
```

リスト9

```
;
; BDOS CALL #31 ( Get Addr (Disk parms) )
;
        CSEG
        ORG     100H                          8080モデル

        MOV     AX,CS    ! MOV   DS,AX

        MOV     CL,09H
        MOV     DX,OFFSET MESSAGE1            "BDOS CALL…"の表示
        INT     224

        MOV     CL,09H
        MOV     DX,OFFSET MESSAGE2            "Disk Parameter…"の表示
        INT     224

        MOV     CL,31
        INT     224                           DPBのアドレスを得る

        CALL    PRINT_DPB                     DPBを表示

        XOR     CL,CL
        XOR     DL,DL                         CCPへ
        INT     224

PRINT_DPB:
        MOV     DX,OFFSET SPT
        CALL    PRINT_STRING

        MOV     AX,ES:[BX]
        INC     BX
        INC     BX                            DPBを表示するルーチン

        CALL    PRINT_AX_IN_HEX

        MOV     DX,OFFSET BSH
        CALL    PRINT_STRING

        MOV     AL,ES:[BX]
        INC     BX

        CALL    PRINT_AL_IN_HEX


        MOV     DX,OFFSET BLM
        CALL    PRINT_STRING

        MOV     AL,ES:[BX]
        INC     BX

        CALL    PRINT_AL_IN_HEX


        MOV     DX,OFFSET EXM
        CALL    PRINT_STRING

        MOV     AL,ES:[BX]
        INC     BX

        CALL    PRINT_AL_IN_HEX


        MOV     DX,OFFSET DSM
        CALL    PRINT_STRING

        MOV     AX,ES:[BX]
        INC     BX
        INC     BX

        CALL    PRINT_AX_IN_HEX


        MOV     DX,OFFSET DRM
        CALL    PRINT_STRING

        MOV     AX,ES:[BX]
        INC     BX
        INC     BX

        CALL    PRINT_AX_IN_HEX


        MOV     DX,OFFSET AL0
        CALL    PRINT_STRING

        MOV     AL,ES:[BX]
        INC     BX

        CALL    PRINT_AL_IN_HEX

        MOV     DX,OFFSET AL1
        CALL    PRINT_STRING

        MOV     AL,ES:[BX]
        INC     BX

        CALL    PRINT_AL_IN_HEX


        MOV     DX,OFFSET CKS
```

30 ファイル属性のセット

Set File Attributer

使い方

MOV CL, 1EH
MOV DX, FCBのオフセット
INT 224
AL …0 正常終了
…FFH ファイルは存在せず

「ファイル属性のセット」はCP/M-86のファイル属性である，

R/O⟷R/W, DIR(普通のファイル)⟷SYS(見えないファイル),
Archive(CP/M-86では未使用)

図C ディスクパラメータブロック(DPB)

SPT	BSH	BLM	EXM	DSM	DRM	AL0	AL1	CKS	OFF
16ビット	8ビット	8ビット	8ビット	16ビット	16ビット	8ビット	8ビット	16ビット	16ビット

ディスクパラメータブロック(DPB)の項目

フィールド	内容
SPT	トラック当たりの全セクタ数
BSH	データアロケーションブロックシフトファクタであって，データブロックの割り当ての大きさによって決まってきます。
BLM	データブロックの割り当ての大きさによって決まってくるブロックマスクです。
EXM	エクステンドマスクであって，データブロックの割り当ての大きさとディスクのブロック数によって決まります。
DSM	ディスクの全容量を決めます。
DRM	このディスクに格納できるディレクトリエントリの総数について決めます。
AL0, AL1	確保するディレクトリブロックを決めます。
CKS	ディレクトリのチェックベクトルの大きさになります。
OFF	論理フロッピディスクの先頭にある確保済みのトラックの数になります。

をファイルにセットします。セットの方法は，FCB内の t1～t3 の最上位ビットをセットするかどうかによります。

t1のMSB　0…R/W　1…R/O
t2　〃　0…DIR　1…SYS
t3　〃　0　1…Archive

リスト8に「ファイル属性のセット」の例を示します（実行例**図8**）。

31 ディスクパラメータアドレスの取り出し

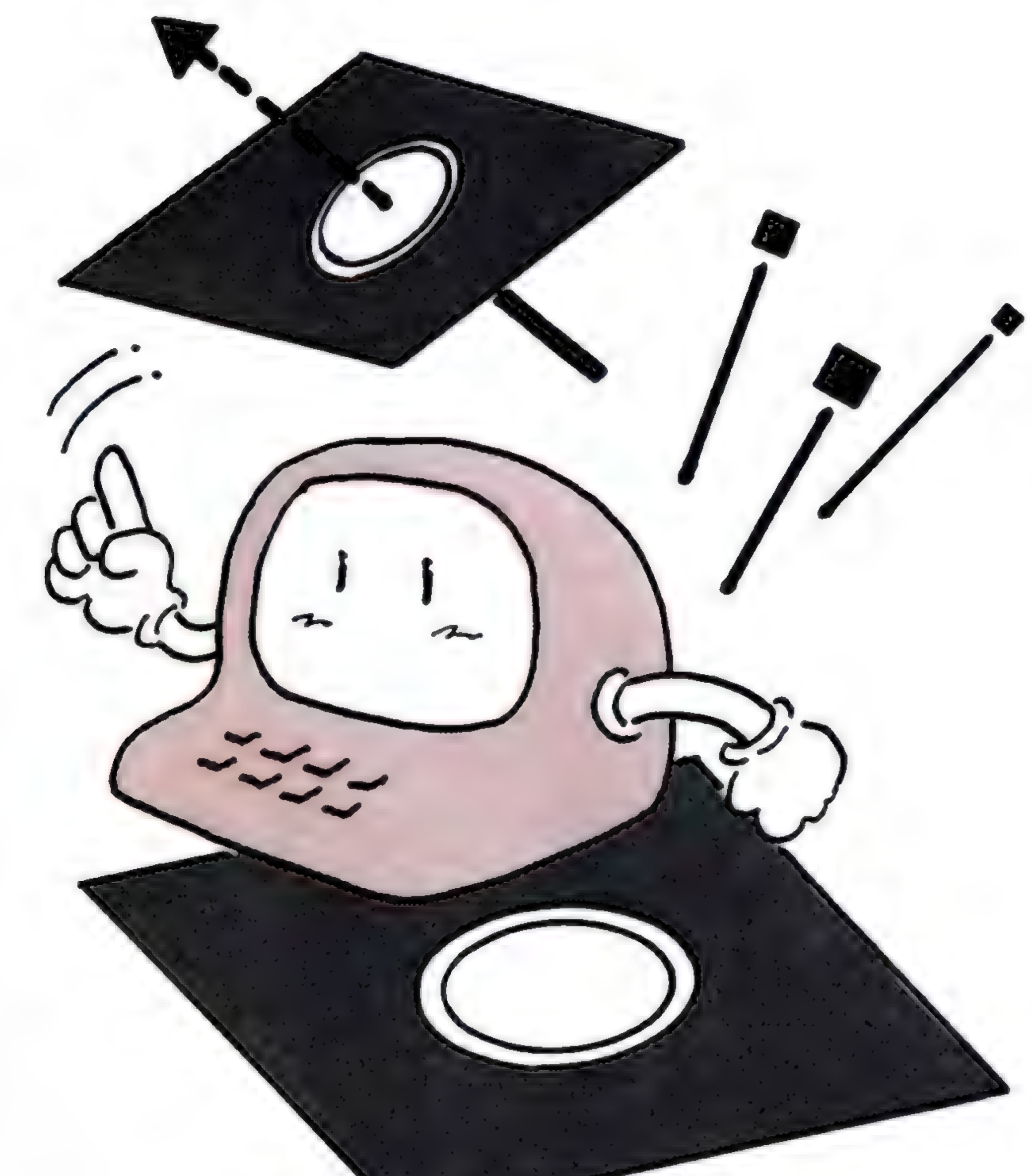
Get Addr (Disk Parms)

使い方

MOV　CL，1FH
INT　224
BX　：　DPBのオフセット ｝が返る
ES　：　セグメントベース

「ディスクパラメータアドレスの取り出し」は，BIOS 内のディスクパラメータブロック(DPB)の先頭アドレスを返します。DPB の構成は**図C**のようになっています。図のように DPB はドライブの諸元を設定しています。

これによりディスクの容量などを調べたりすることもできます。**リスト9**に DPB を表示する例を示します。



```
        CALL    PRINT_STRING

        MOV     AX,ES:[BX]
        INC     BX
        INC     BX

        CALL    PRINT_AX_IN_HEX


        MOV     DX,OFFSET OFF
        CALL    PRINT_STRING

        MOV     AX,ES:[BX]
        INC     BX
        INC     BX

        CALL    PRINT_AX_IN_HEX

        RET
PRINT_STRING:
        PUSH    ES
        PUSH    BX
        MOV     CL,9
        INT     224
        POP     BX
        POP     ES
        RET

PRINT_AX_IN_HEX:
        PUSH    ES
        PUSH    BX

        PUSH    AX
        MOV     AL,AH
        CALL    PRINT_AL
        POP     AX
        CALL    PRINT_AL                AXを16進表示するルーチン
        MOV     CL,2
        MOV     DL,'H'
        INT     224

        POP     BX
        POP     ES
        RET

PRINT_AL_IN_HEX:
        PUSH    ES
        PUSH    BX

        CALL    PRINT_AL
        MOV     CL,2
        MOV     DL,'H'                  ALを16進表示するルーチン
        INT     224

        POP     BX
        POP     ES
        RET
PRINT_AL:
        MOV     AH,AL
        AND     AL,0F0H
        MOV     CL,4
        SHR     AL,CL
        CALL    PRINT_AL0

        MOV     AL,AH
        AND     AL,0FH
        CALL    PRINT_AL0

        RET
PRINT_AL0:
        CMP     AL,10   ! JAE   PRINT_AL1
        ADD     AL,'0'
        JMPS    PRINT_AL2
PRINT_AL1:
        ADD     AL,'A'-10
PRINT_AL2:
        PUSH    AX
        MOV     DL,AL
        MOV     CL,2
        INT     224
        POP     AX
        RET

END_CS:
        DSEG
        ORG     OFFSET END_CS
MESSAGE1        DB      'BDOS CALL #31 ( Get Addr(Disk Parm) )',0DH;0AH,
'$'
MESSAGE2        DB      'Disk Parameter Block: ',0DH,0AH,'$'
SPT             DB      '   SPT:$'
BSH             DB      '   BSH:$'
BLM             DB      '   BLM:$'
EXM             DB      '   EXM:$'
DSM             DB      '   DSM:$'
DRM             DB      '   DRM:$'
AL0             DB      '   AL0:$'
AL1             DB      '   AL1:$'
CKS             DB      '     CKS:$'
OFF             DB      '   OFF:$'

        END
```

リスト10

```
;
; BDOS CALL #32 ( Get/Set User Code )
;
        CSEG
        ORG     100H                              8080モデル
                                                  
        MOV     AX,CS   ! MOV   DS,AX             

        MOV     CL,09H                            "BDOS CALL…"の表示
        MOV     DX,OFFSET MESSAGE1
        INT     224

        MOV     CL,09H                            "User Code:"の表示
        MOV     DX,OFFSET MESSAGE2
        INT     224

        MOV     CL,32                             User Codeを得る
        MOV     DL,0FFH
        INT     224

        CALL    PRINT_AL_IN_HEX                   ALレジスタを16進表示

        XOR     CL,CL                             CCPへ
        XOR     DL,DL
        INT     224

PRINT_AL_IN_HEX:                                  ALレジスタを16進表示するルーチン
        PUSH    ES
        PUSH    BX

        CALL    PRINT_AL
        MOV     CL,2
        MOV     DL,'H'
        INT     224

        POP     BX
        POP     ES
        RET
PRINT_AL:
        MOV     AH,AL
        AND     AL,0F0H
        MOV     CL,4
        SHR     AL,CL
        CALL    PRINT_AL0

        MOV     AL,AH
        AND     AL,0FH
        CALL    PRINT_AL0

        RET
PRINT_AL0:
        CMP     AL,10   ! JAE   PRINT_AL1
        ADD     AL,'0'
        JMPS    PRINT_AL2
PRINT_AL1:
        ADD     AL,'A'-10
PRINT_AL2:
        PUSH    AX
        MOV     DL,AL
        MOV     CL,2
        INT     224
        POP     AX
        RET

END_CS:
        DSEG
        ORG     OFFSET END_CS
MESSAGE1        DB      'BDOS CALL #32 ( Set/Get User Code )',0DH,0AH,'$

MESSAGE2        DB      'User Code : $'

        END
```

図9

```
A>
A>USER 0        ユーザ番号を0に
A>
A>OHFM32
BDOS CALL #32 ( Set/Get User Code )
User Code : 00H
A>              ──────────→ 確かに0
A>PIP A:[G2]=OHFM32.CMD      2番にOHFM32.CMDを転送

A>
A>USER 2        ユーザ番号を2に
A>
A>OHFM32
BDOS CALL #32 ( Set/Get User Code )
User Code : 02H
A>              ──────────→ 確かに2になっている
A>
```

32 ユーザコードのセット/取り出し

Set/Get User Code

使い方

	MOV	CL, 20H
(Get)	MOV	DL, 0FFH
	INT	224
	AL	ユーザコード
(Set)	MOV	DL, ユーザコード
	INT	224

「ユーザコードのセット，取り出し」は，ユーザ番号をセットしたり返したりを行います。ユーザ番号は同一のディスクを多人数で使う場合などに有効ですが，CP/M-86ではあまり意味はないでしょう。

ユーザ番号は0～15ですから，DLレジスタには0～15をセットすべきですが，CP/M-86では，0FHとAND をとっているようなので，16＝0，17＝1，18＝2……となるようです。

リスト10に，このBDOSコールの例を示します（実行例図9）。

ユーザ番号は，USERコマンドでセットしたり，表示したりすることができますから，このプログラムの効果はUSERコマンドで調べることができます。

今回はBDOSコールの23(Rename File)から32(Set/Get User Code) までを説明しました。

BDOSコールのすべての機能を知っている必要はないと思います。たとえば，DPBの構造がどうなっているかなどは，大半のユーザには関係のないことです。しかし，これはBIOS 部をすべて自分で書く人間にとっては非常に重要なことです。また，なぜCP/M-86が異なるハードウェア上で動作しているのかを理解するには知っておくべきことでしょう。そんなこともあるのかと，頭のすみにでも覚えておいてください。

簡易ジョイスティックの製作

傍若 無人

キーボードなどの改造なし，ジャンク品なら500円でできるジョイスティックを紹介します。スイッチとダイオードだけの，だれにでも簡単に作れるジョイスティックです。

はじめに

FMシリーズでは，リアルタイムゲームがおもしろくない。キー操作が難しい。キーが1度押されると，他のキーが押されるまで信号が変わらない。そのため，FMシリーズのリアルタイムゲームは，ホントのリアルタイムゲームとは言えない。押している間だけ進み，離せば止まる，でなければリアルタイムではない。このキー入力の特殊性を解消するため，キーボードを改造したジョイスティックなどもあるが，改造となると，ちょっとしりごみしてしまう。PIAなどを外付けすればよいのだろうが，それも面倒だ。なにかいい手はないかといろいろやってみたあげく，プリンタポートにジョイスティックをつないでみようと思いついた。

では，簡易ジョイスティック製作講座ハード編・ソフト編・応用ゲームと進めてみよう。

ハード編

ハードといっても電気部品はスイッチ，ダイオード，コネクタ，リード線少々と中学生の技術の授業並なので，だれでも簡単にできると思う。凝るとすればケースやスイッチの配列ぐらいなものだが，これが結構ゲームに影響するので，自分の納得いくものに，各自工夫してもらおう。

回路図を**図1**に示す。全体の回路としてはよく使うマトリックスの構造である。使用ダイオードは，スイッチング用のもので順方向電圧の低いもの(1S1588など）を用いてもらいたい。順方向電圧が高い場合は，**図2**に示すような不都合な点がでてくるので気をつけてほしい。コンピュータ側とのインタフェースは実に簡単で，データバスでスイッチの選択を行い，PE（ペーパエンプティ）と ＊ERROR（プリンタエラー）をスイッチが押されたかどうかの入力に用いている。FM-7,8では，プリンタの種類を判別するためにL1，L2というもう2ビ

図1 回路図

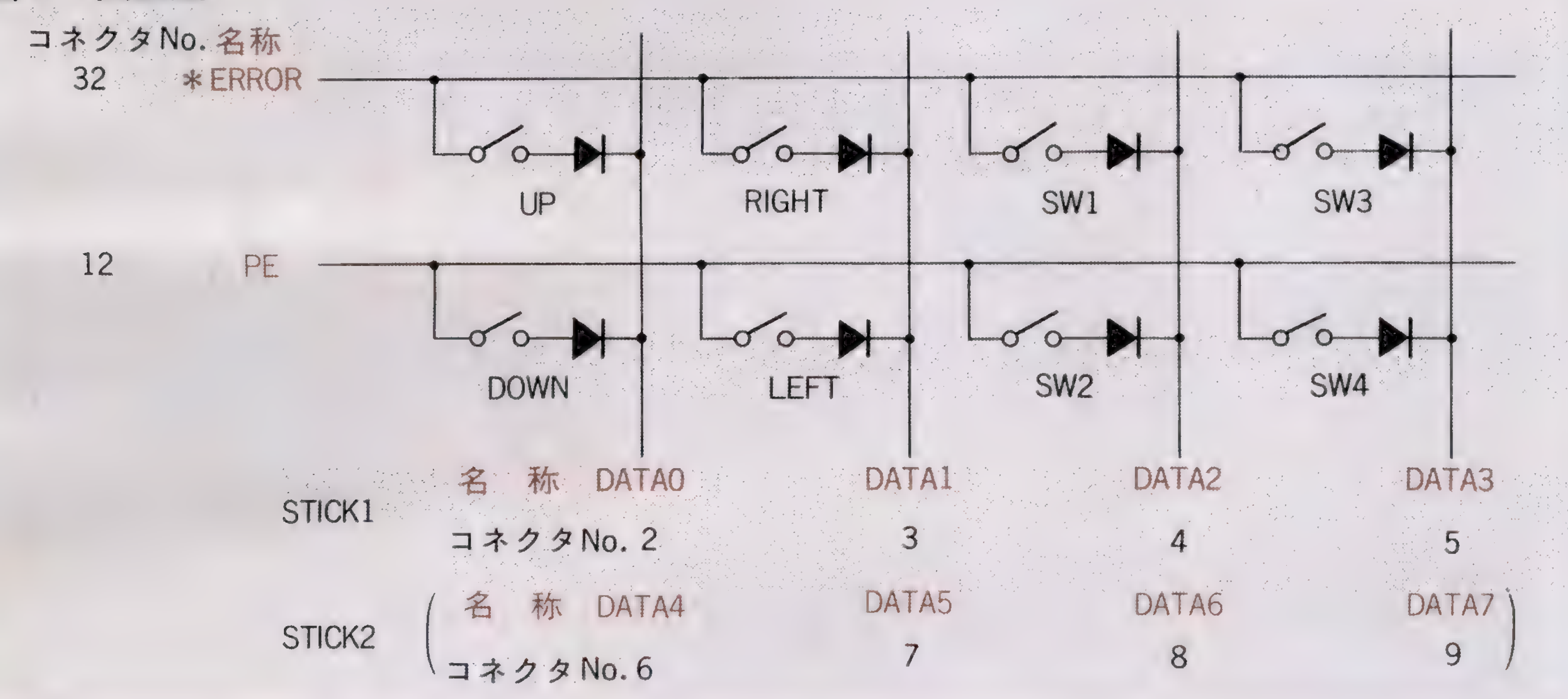
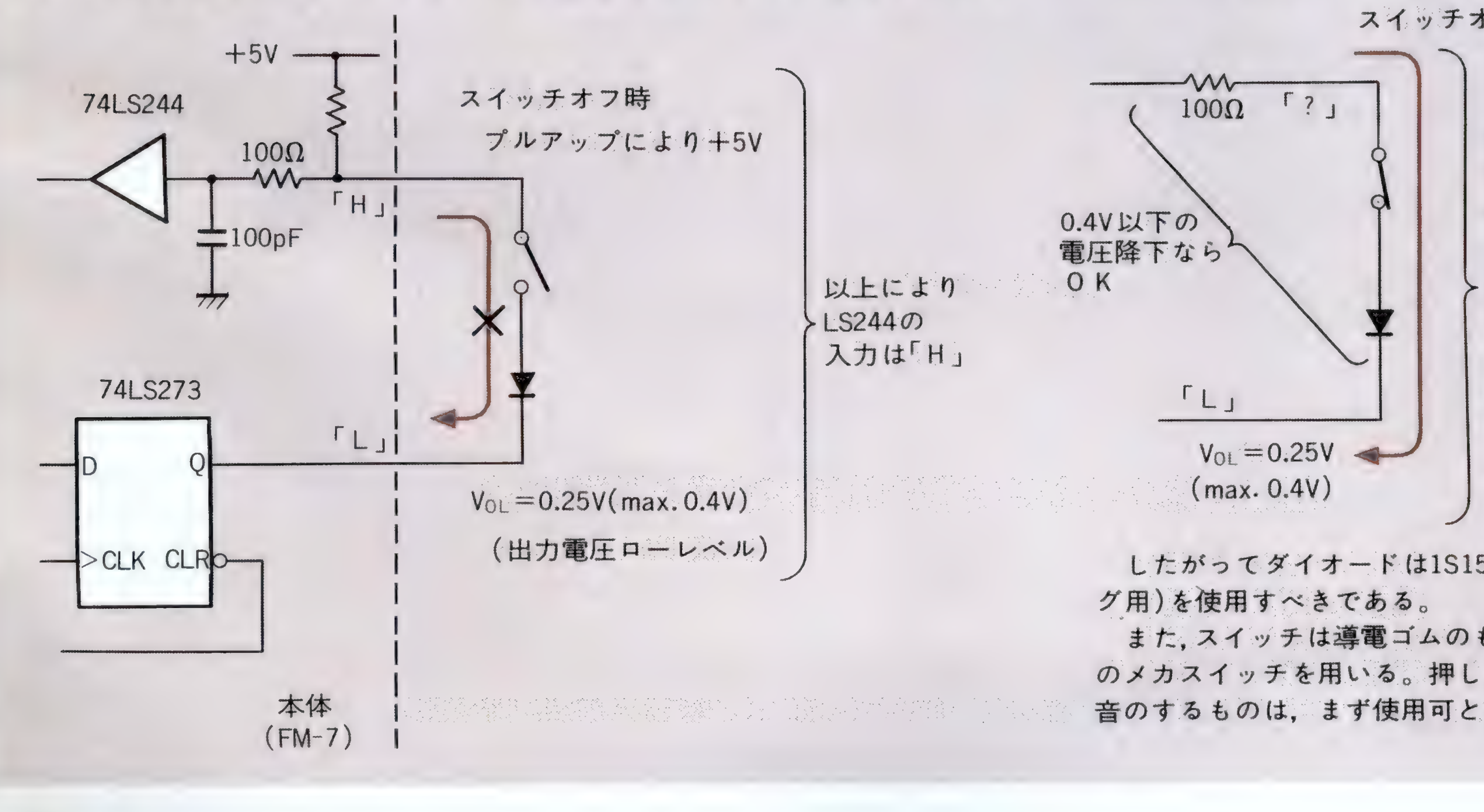


図2 順方向電圧の高いダイオードや導電ゴムスイッチを用いたとき(入力と出力間に電位差が生じる場合)



LS244のI_{IL}(入力電流ローレベル）は0.2mAと微小である。一方，V_{IL}(入力電圧ローベル）は0.8V以下である必要がある(通常0.4V)。そこで，スイッチ＋ダイオードによる電圧降下は0.8－0.4＝0.4V以下にしなければならない。ところが導電ゴムスイッチや高逆耐圧のダイオードの中には，微小電流で大きな電圧降下を生じるものがあり，LS244がうまく「L」にならない場合がある。

したがってダイオードは1S1588など論理演算用のSW(スイッチング用)を使用すべきである。

また，スイッチは導電ゴムのものは避けて，マイクロスイッチなどのメカスイッチを用いる。押したときに，「カチッ」という固い金属音のするものは，まず使用可とみてよい。

表1　ジョイスティック1台

	部品名	形式	個数	単価
配線	マイクロスイッチ	SS-5GL	4	100
	プッシュスイッチ	TR-3	4	100
	ダイオード	1S1588sw	8	20
	プリンタコネクタ*	アンフェノール57-40360	1	500
	ケース	YM-150	1	680
	リード線		適量	
メカ	ネジ	2mm φ×30mm	8	
	スプリングワッシャ	2mm φ	8	
	ナット	2mm φ	8	
	ジュラコンスペーサ	20mm	8	
	ネジ	3mm φ×50mm	1	15
	ジュラコンスペーサ	15mm	1	
	ジュラコンスペーサ	10mm	1	
	スプリング	3mm φネジの入るもの	1	
	カラーナット	3mm φ×30mm	1	30
	平ワッシャ	3mm φ	6	
	スプリングワッシャ	3mm φ	5	
	ナット	3mm φ	5	
	ネジ	3mm φ×10mm	5	

●高いと思う人はケース・コネクタを安くあげましょう。

●価格はお店によって多少バラツキがあります。

*プリンタケーブルがなく，直づけにするときは，FM-7,8の場合は富士通製J034-01かスリーエムの34ピンコネクタ，FM-11,77の場合はアンフェノールDDK57-30360にて本体に接続。

表2　ジョイスティック2台

●表1の部品×2
(ただしプリンタコネクタのみ1個)

部品名	形名	個数	単価
6ピンDIN コネクタ	MAB-6	2	160
6ピンDIN ソケット	MAS-60	2	220
6ピン無線マイク用カールコード		3mm	600

図4　ケース穴あけ寸法［単位：mm］

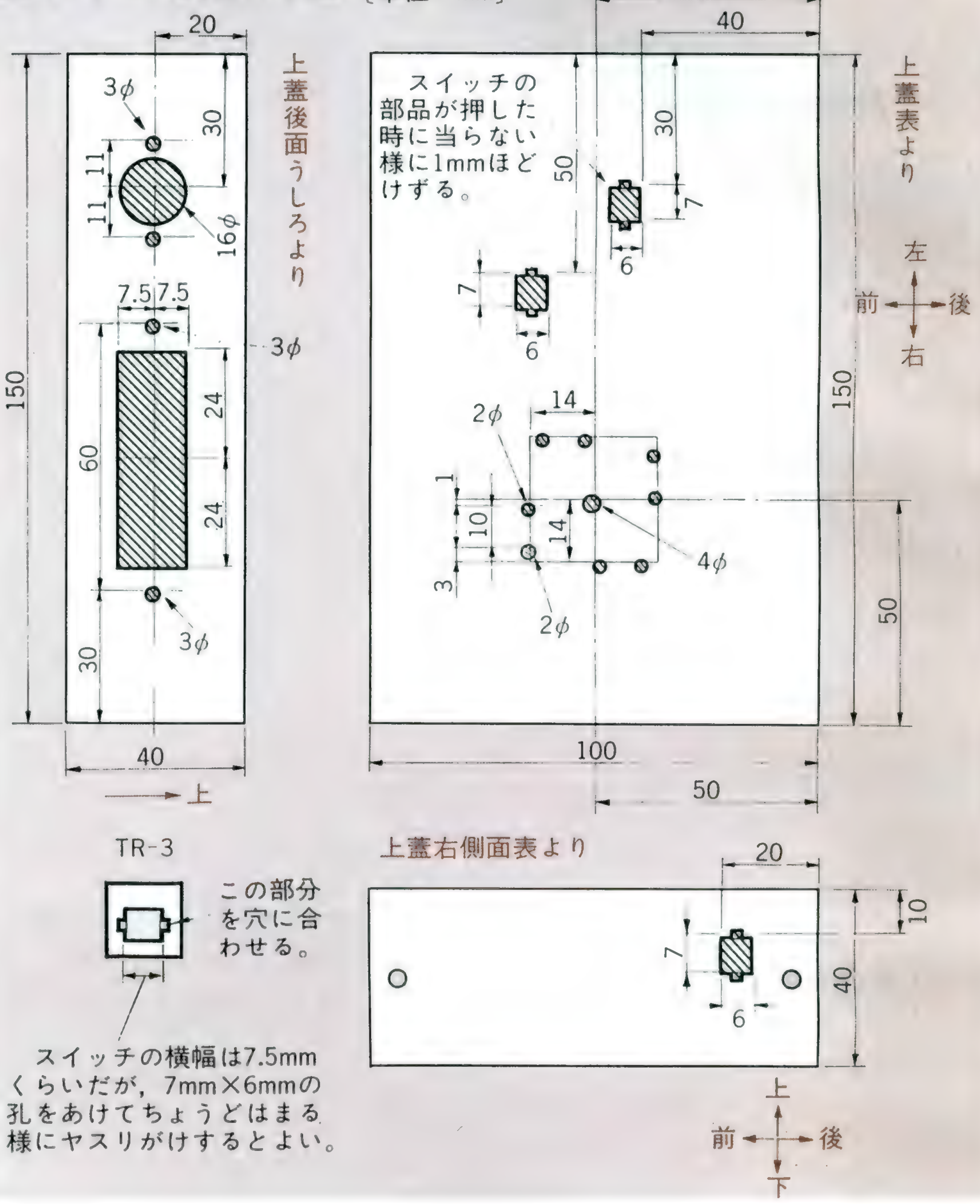
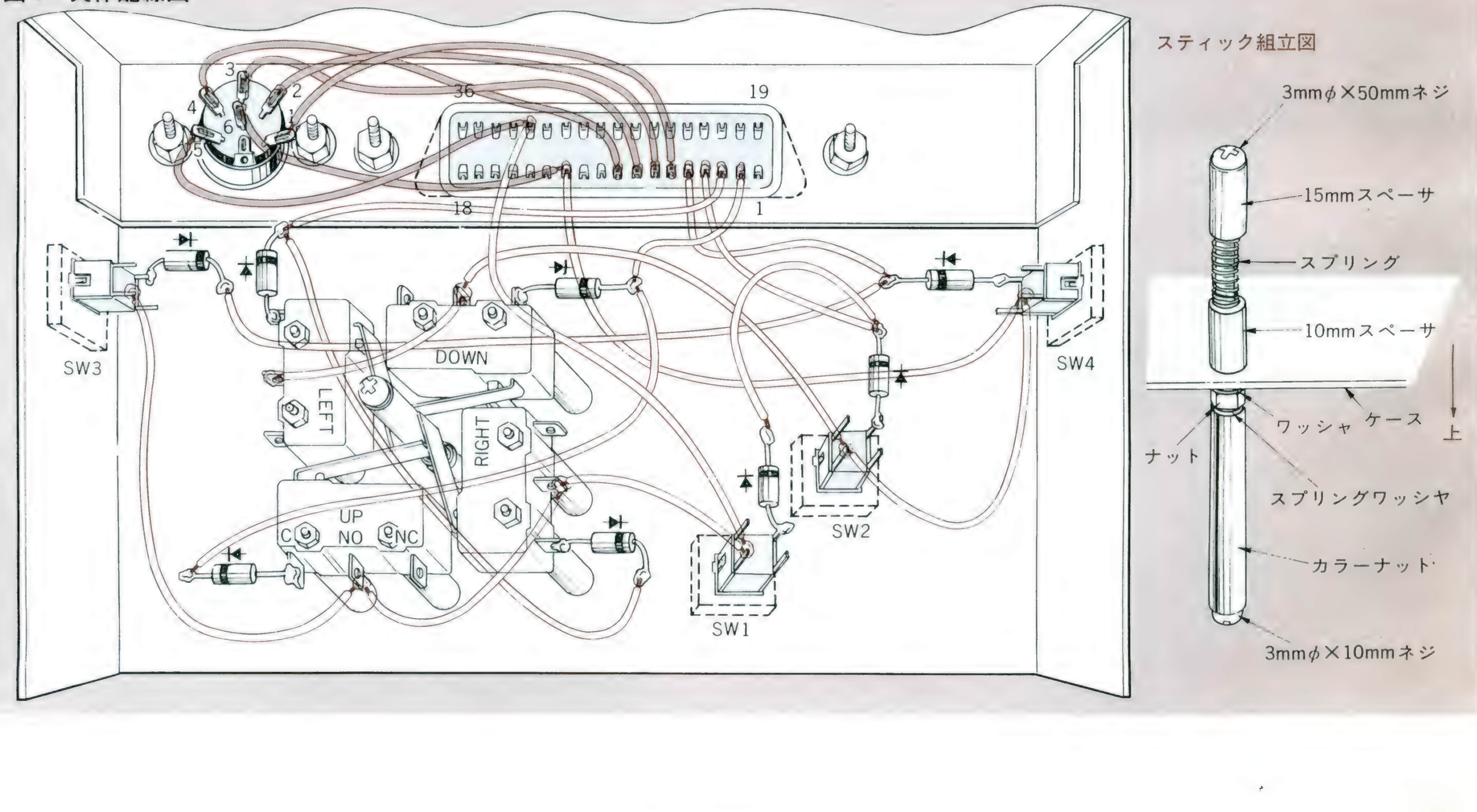
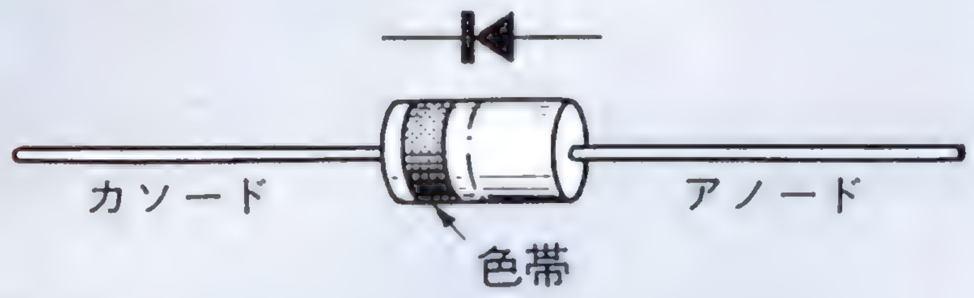


図3　実体配線図

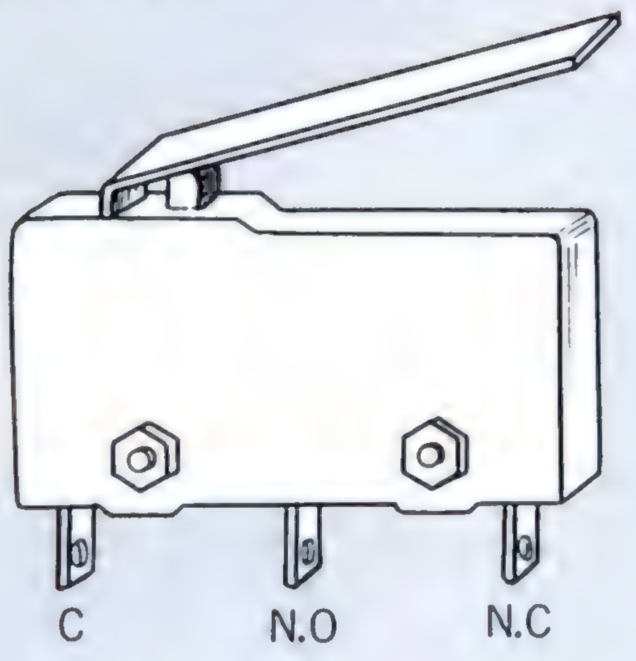


部品について

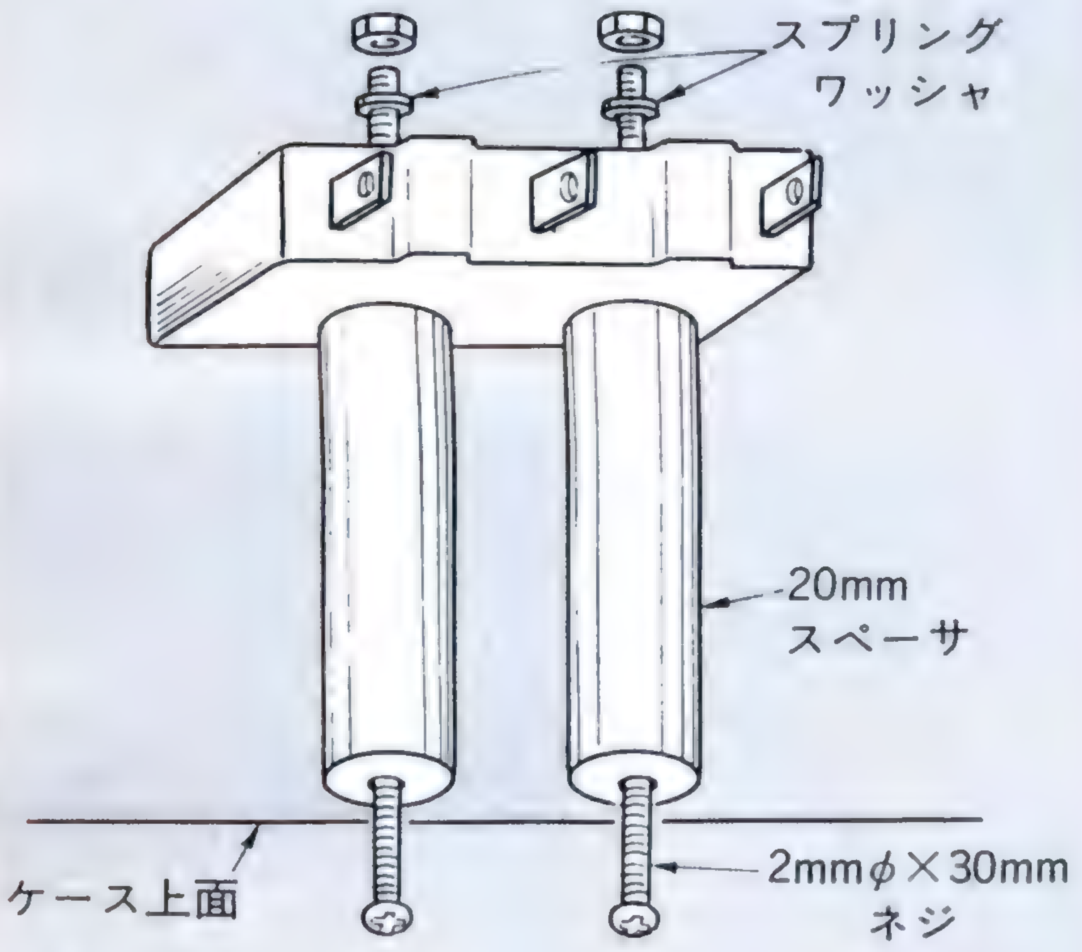
● ダイオード1S1588(図2参照)



● マイクロスイッチSS-5GL

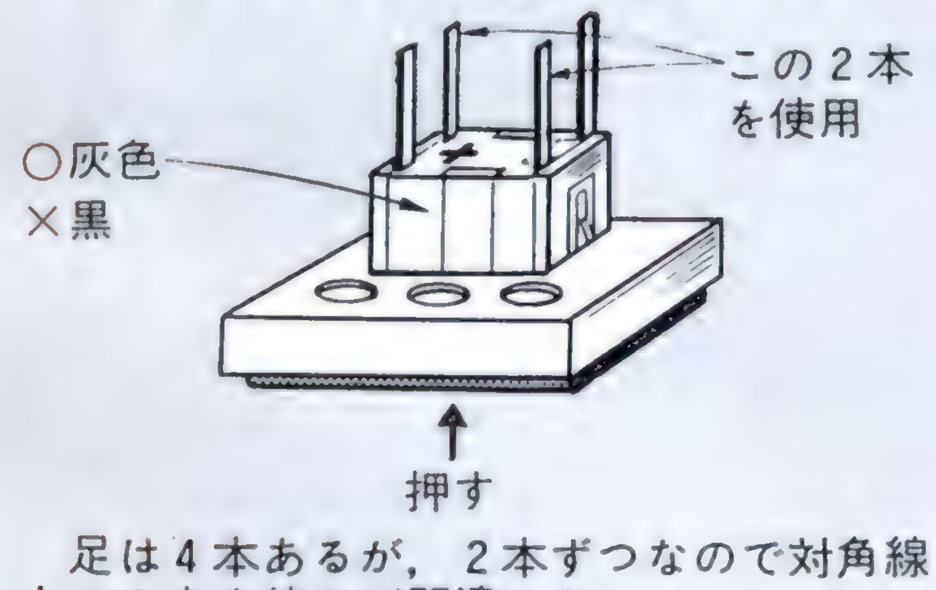


マイクロスイッチとケース(上面)との間にはスペーサを入れて，テコの原理でジョイスティックを使う。



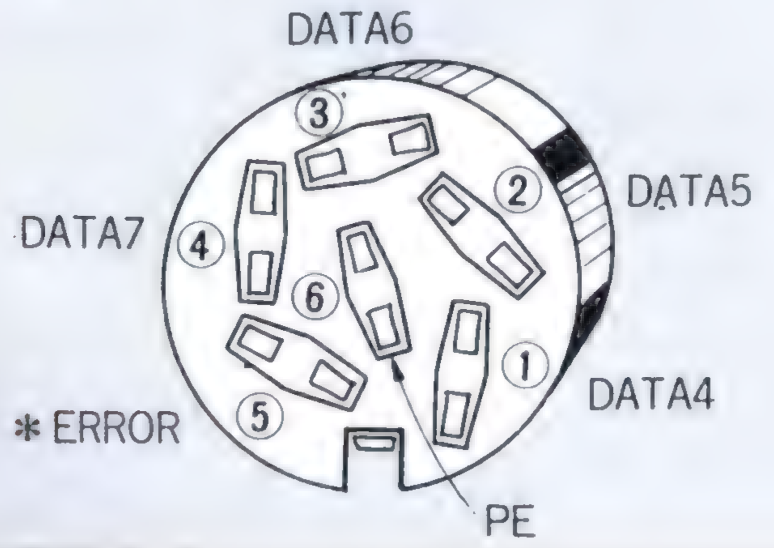
● プッシュスイッチ(TR-3)

TR-3という型番で，導電ゴムのものもあるので注意が必要。こちらで手に入れたものは，「黒」が導電ゴムであったが，押して「カチッ」と音がするか確認してほしい。



足は4本あるが，2本ずつなので対角線上の2本を使えば間違いない。

● 6ピンDIN



ットの入力が用意されている。しかし,FM-11ではその機能が変更されているので，今回は使用しなかった。またBUSY(プリンタビジー)とACK(アクノレッジ)は，IRQの関係もあり用いなかった。以上より，多少スピードの遅れとソフトの面倒さが生じるが，FM-7,8,11すべてに使用できるし，充分実用になると思われる。

部品表を**表1,2**に，実体配線図を**図3**，ケース穴あけ寸法を**図4**に示す。前述のように，順方向電圧の低いダイオードとメカスイッチを選べばOKなので，**写真1**のように，それぞれ使いやすい形で作ってほしい。

プリンタポートとの接続は，プリンタケーブルがあるものとして部品表などに示したが,持っていない場合は,この際，プリンタケーブルのみ買ってしまうか，**図5**，**表3**のようにプリンタポートから直接引き出してもらいたい。

写真1　ジョイスティック上部裏にマイクロスイッチを取り付け，親ゆびと人差ゆびのあたる部分にSW1,2を取り付けたもの(左)と，プッシュスイッチもマイクロスイッチにしたもの(右)。

図5　本体プリンタコネクタ

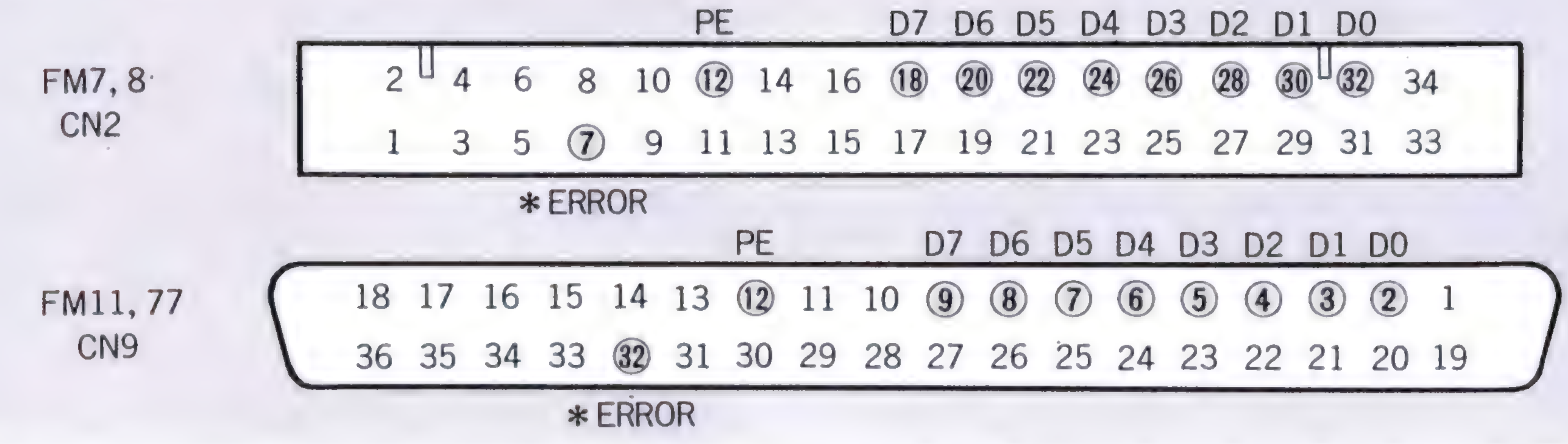


表3

ポート名	入出力	コネクタNo.	FM-11,77本体コネクタNo.	FM-7,8本体コネクタNo.	使用法	
DATA0	OUT	2	2	32	UP or DOWN	ジョイスティック1
DATA1	OUT	3	3	30	RIGHT or LEFT	
DATA2	OUT	4	4	28	SW1,SW2	
DATA3	OUT	5	5	26	SW3,SW4	
DATA4	OUT	6	6	24	UP or DOWN	ジョイスティック2
DATA5	OUT	7	7	22	RIGHT or LEFT	
DATA6	OUT	8	8	20	SW1,SW2	
DATA7	OUT	9	9	18	SW3,SW4	
PE	IN	12	12	12	UP, RIGHT, SW1, SW3の入力	
＊ERROR	IN	32	32	7	DOWN, LEFT, SW2, SW4の入力	

©1984 B.BOUZYAKU

リスト1

```
100 DIM DIR(15)           ── ジョイスティック→キーマトリックス変換テーブル用配列
110 POUT=&HFD01:PIN=&HFD02  ── プリンタ出力ポート＝&HFD01, プリンタ入力＝&HFD02
120 POKE POUT,255         ── プリンタ出力ポート初期化
130 RESTORE 150
140 FOR A=0 TO 15:READ DIR(A):NEXT A  ┐── ジョイスティック→キーマトリックス変換テーブル作成
150 DATA 5,8,2,5,6,9,3,5,4,7,1,5,5,5,5,5  ┘
160 A=0:GOSUB 9000:PRINT A;B;" : ";:A=1:GOSUB 9000:PRINT A;B  ┐── 動作チェック
170 GOTO 160  ┘
9000 ´JOYINSUB  A=0 OR 1-(A,B,C,PIN,POUT)->A=dir(1-9):B=SW
9010 C=2^((A AND 1)*4)    ── ジョイスティック1，2の選択
9020 POKE POUT,(255-C):D=255-(PEEK(PIN)):A=(D AND 8)¥4+(D AND 2)¥2  ── UP, DOWN取り込み
9030 POKE POUT,(255-C*2):D=255-(PEEK(PIN)):A=(D AND 8)+(D AND 2)*2+A  ── RIGHT, LEFT取り込み
9040 POKE POUT,(255-C*4):D=255-(PEEK(PIN)):B=(D AND 8)¥4+(D AND 2)¥2  ── SW1, SW2取り込み
9050 POKE POUT,(255-C*8):D=255-(PEEK(PIN)):B=(D AND 8)+(D AND 2)*2+B  ── SW3, SW4取り込み
9060 A=DIR(A):RETURN      ── キーマトリックス
```

リスト2（リスト2以下は，FM-11BSでは動きません）

```
                                 OPT   PAGE=54
5000                             ORG   $5000
          FD01         LPTOUT    EQU   $FD01     printer out port
          FD02         LPTIN     EQU   $FD02     printer status port
                       *
5000 34   06           ENTRY     PSHS  D         usr function call here
5002 A6   03                     LDA   3,X
5004 8D   06    500C             BSR   JOYIN
5006 A7   03                     STA   3,X
5008 E7   02                     STB   2,X
500A 35   86                     PULS  D,PC
                       *
500C 34   10           JOYIN     PSHS  X         a=0 or 1 -> a=dir(1-9),b=sw
500E C6   EF                     LDB   #%11101111 if a=1 then higher
5010 30   8C 3D                  LEAX  <JOYTAB,PCR
5013 84   01                     ANDA  #1
5015 26   02    5019             BNE   JOY00
5017 C6   FE                     LDB   #%11111110 if a=0 then lower
5019 8D   23    503E   JOY00     BSR   JOYS01    get 2 bit
501B 34   02                     PSHS  A
501D 1A   01                     ORCC  #1
501F 59                          ROLB
5020 8D   1C    503E             BSR   JOYS01    get 2 bit
5022 48                          LSLA
5023 48                          LSLA
5024 AA   E0                     ORA   ,S+       joystick data
5026 A6   86                     LDA   A,X       dir->tenkey matrix
5028 34   02                     PSHS  A
502A 1A   01                     ORCC  #1
502C 59                          ROLB
502D 8D   0F    503E             BSR   JOYS01    get 2 bit
502F 34   02                     PSHS  A
5031 1A   01                     ORCC  #1
5033 59                          ROLB
5034 8D   08    503E             BSR   JOYS01    get 2 bit
5036 48                          LSLA
5037 48                          LSLA
5038 AA   E0                     ORA   ,S+       switch data
503A 1F   89                     TFR   A,B
503C 35   92                     PULS  A,X,PC
503E F7   FD01         JOYS01    STB   LPTOUT
5041 B6   FD02                   LDA   LPTIN
5044 43                          COMA
5045 46                          RORA
5046 46                          RORA
5047 34   01                     PSHS  CC
5049 46                          RORA
504A 35   01                     PULS  CC
504C 49                          ROLA
504D 84   03                     ANDA  #%00000011
504F 39                          RTS
```

ソフト編

ハードのところで述べたように，回路は単純なマトリックスなので，スイッチのセレクトと状態の入力を繰り返してやるだけでよい。今回は，入力ポートを2ビットしか使用しなかったので，ジョイスティックからの入力を2回，スイッチ4つで2回と計4回（1人分で）のスキャンが必要となる。ジョイスティックの入力は，テーブルを用いて，テンキーの1から9に置き換えておけば，すでにあるBASICのゲームにも応用しやすいと考えた。BASICによるコントロールプログラムを**リスト1**に，アセンブラによるソースリストを**リスト2**に，動作チェックとUSR0によるサポートを**リスト3**に示す。このプログラムは，ポジションインディペンデントで，どのアドレスにも置

くことができる。**リスト4**は機械語のテストプログラムである。使い方，データの形式も，注釈として加えた。プログラムの高速化と短縮のために，チャタリングに対する処理は行っていないが，マイクロスイッチなどを用いれば，ほとんど動作に問題ないようである。

```
                               *
       5050     05             JOYTAB FCB    5
       5051     08                    FCB    8
       5052     02                    FCB    2
       5053     05                    FCB    5
       5054     06                    FCB    6
       5055     09                    FCB    9
       5056     03                    FCB    3
       5057     05                    FCB    5
       5058     04                    FCB    4
       5059     07                    FCB    7
       505A     01                    FCB    1
       505B     05                    FCB    5
       505C     05                    FCB    5
       505D     05                    FCB    5
       505E     05                    FCB    5
       505F     05                    FCB    5
                5000                  END    ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=5000
PROGRAM END   ADDR=505F
PROGRAM ENTRY ADDR=5000
```

リスト3

```
100 CLEAR 500,&H4FFF:CLS
110 GOSUB 9000 ——— USR0の作成と設定
120 J0=USR0(0):J1=USR0(1)
130 LOCATE 0,0:PRINT USING"JOY1 : DIR=# SW1=# SW2=# SW3=# SW4=#  ";J0 AND &HF;J0
 ¥ &H100 AND 1;J0 ¥ &H200 AND 1;J0 ¥ &H400 AND 1;J0 ¥ &H800 AND 1;
140 PRINT USING"JOY2 : DIR=# SW1=# SW2=# SW3=# SW4=#  ";J1 AND &HF;J1 ¥ &H100 AN
D 1;J1 ¥ &H200 AND 1;J1 ¥ &H400 AND 1;J1 ¥ &H800 AND 1
150 GOTO 120
9000 RESTORE 9030:A=&H5000:DEFUSR0=A
9010 READ A$:IF A$="@" THEN RETURN
9020 POKE A,VAL("&H"+A$):A=A+1:GOTO 9010
9030 DATA 34,06,A6,03,8D,06,A7,03
9040 DATA E7,02,35,86,34,10,C6,EF
9050 DATA 30,8C,3D,84,01,26,02,C6
9060 DATA FE,8D,23,34,02,1A,01,59
9070 DATA 8D,1C,48,48,AA,E0,A6,86
9080 DATA 34,02,1A,01,59,8D,0F,34
9090 DATA 02,1A,01,59,8D,08,48,48
9100 DATA AA,E0,1F,89,35,92,F7,FD
9110 DATA 01,B6,FD,02,43,46,46,34
9120 DATA 01,46,35,01,49,84,03,39
9130 DATA 05,08,02,05,06,09,03,05
9140 DATA 04,07,01,05,05,05,05,05
9150 DATA 00,00,00,00,00,00,00,00
9160 DATA @ ——— データエンドフラグ
```

（130～140：動作チェック）

（9030～9150：リスト2のオブジェクト）

このプログラムは，ポジションインディペンデントなので□の部分を変更すれば，好きなところにおくことができる。

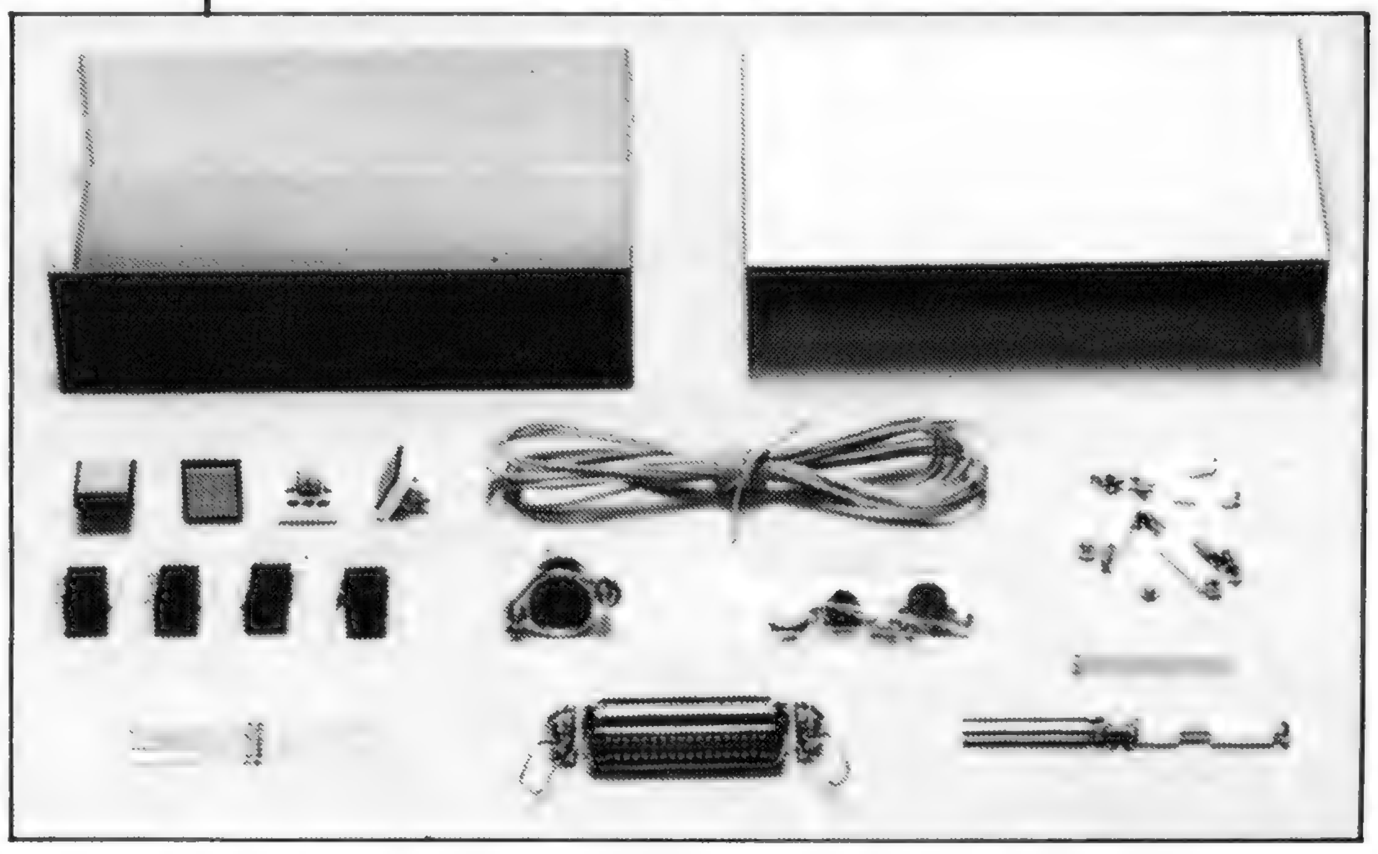
&H4FFF ——→置きたいアドレスの先頭－1

&H5000 ——→置きたいアドレスの先頭番地

リスト4

```
                          *
                          *        JOYSTICK SAMPLE
                          *
                                 NAM    STICKTEST
                                 OPT    S,NOGEN,NOMEM,OBJ
             FD01      LPTOUT EQU    $FD01    printer out port
             FD02      LPTIN  EQU    $FD02    printer status port
             FBFA      BIOS   EQU    $FBFA
      4F00                    ORG    $4F00
      4F00   10        RQNO   FCB    16       SUBOUT
      4F01   00        RCBSTA FCB    0
      4F02   4F08      RCBDBA FDB    LINBUF
      4F04   000E      RCBLN  FDB    14
      4F06   0000             FDB    0
      4F08   0000      LINBUF FDB    0
      4F0A   15               FCB    21       LINE
      4F0B   00        LCOL   FCB    0
      4F0C   00        LFUNC  FCB    0
      4F0D   0000      LX1    FDB    0
```

写真2　部品



```
4F0F      0000      LY1     FDB    0
4F11      0000      LX2     FDB    0
4F13      0000      LY2     FDB    0
4F15      01        BFLAG   FCB    1         BOX
4F16      0000      COL     FDB    0
4F18      00        TRIG    FCB    0
          004F              SETDP  $4F
5000                        ORG    $5000
5000 34   7F        ENTRY   PSHS   CC,D,DP,X,Y,U
5002 86   4F                LDA    #$4F
5004 1F   8B                TFR    A,DP
5006 4F             CSCRN   CLRA
5007 5F                     CLRB
5008 97   18                STA    TRIG
500A DD   0D                STD    LX1
500C DD   0F                STD    LY1
500E CC   027F              LDD    #639
5011 DD   11                STD    LX2
5013 CC   00C7              LDD    #199
5016 DD   13                STD    LY2
5018 CC   0000              LDD    #$0000    COL.0,PSET
501B DD   0B                STD    LCOL
501D 30   8D FEDF           LEAX   RQNO,PCR
5021 AD   9F FBFA           JSR    [BIOS]
5025 CC   0140              LDD    #320
5028 DD   0D                STD    LX1
502A DD   11                STD    LX2
502C CC   0064              LDD    #100
502F DD   0F                STD    LY1
5031 DD   13                STD    LY2
5033 CC   0700              LDD    #$0700    COL.7,PSET
5036 DD   16                STD    COL
5038 DC   16        LOOP    LDD    COL
503A DD   0B                STD    LCOL
503C 30   8D FEC0           LEAX   RQNO,PCR
5040 AD   9F FBFA           JSR    [BIOS]
5044 4F                     CLRA
5045 17   009D 50E5         LBSR   JOYIN
5048 34   07                PSHS   CC,D
504A CC   0000              LDD    #$0000    COL.0,PSET
504D DD   0B                STD    LCOL
504F 30   8D FEAD           LEAX   RQNO,PCR
5053 AD   9F FBFA           JSR    [BIOS]
5057 35   07                PULS   CC,D
5059 29   6F   50CA         BVS    CGCOL     IF SW2=ON THEN CGCOL
505B 34   01                PSHS   CC
505D 0F   18                CLR    TRIG
505F 35   01                PULS   CC
5061 25   32   5095         BCS    CGBOX     ELSE IF SW1=ON THEN CGBOX
5063 9E   0D        MOVE    LDX    LX1
5065 30   86                LEAX   A,X
5067 1E   01                EXG    D,X
5069 4D                     TSTA
506A 2A   03   506F         BPL    MOVE00
506C CC   0000              LDD    #0
506F 1083 0280      MOVE00  CMPD   #640
5073 2B   03   5078         BMI    MOVE01
5075 CC   027E              LDD    #638
5078 1E   01        MOVE01  EXG    D,X
507A 9F   0D                STX    LX1
507C 9E   0F                LDX    LY1
507E 30   85                LEAX   B,X
5080 1E   01                EXG    D,X
5082 4D                     TSTA
5083 2A   03   5088         BPL    MOVE02
5085 CC   0000              LDD    #0
5088 1083 00C8      MOVE02  CMPD   #200
508C 2B   03   5091         BMI    MOVE03
508E CC   00C6              LDD    #198
5091 1E   01        MOVE03  EXG    D,X
5093 9F   0F                STX    LY1
5095 9E   11        CGBOX   LDX    LX2
5097 30   86                LEAX   A,X
5099 1E   01                EXG    D,X
509B 4D                     TSTA
509C 2A   03   50A1         BPL    CGB00
509E CC   0000              LDD    #0
50A1 1083 0280      CGB00   CMPD   #640
50A5 2B   03   50AA         BMI    CGB01
50A7 CC   027E              LDD    #638
50AA 1E   01        CGB01   EXG    D,X
50AC 9F   11                STX    LX2
50AE 9E   13                LDX    LY2
50B0 30   85                LEAX   B,X
50B2 1E   01                EXG    D,X
50B4 4D                     TSTA
50B5 2A   03   50BA         BPL    CGB02
```

応用ゲーム

今までのキーボード入力のゲームでは，キーを離したかどうかの判断と，複数キーの同時入力時の処理が問題となっていたが，ジョイスティックの製作により解決されたと思う。応用ゲームとして，２人ゲームの「バリケード」あるいは「ライナー」，「生き残りゲーム」とか呼ばれている陣取りゲームを作ってみた。ジョイスティックで壁の伸びる方向をコントロールして，お互いの走行を妨害するゲームで，先に壁にぶつかったほうが負けである。映画「トロン」でもおなじみのゲームだ。

スイッチ１を押すことで加速でき，離すと減速するようになっている。速度によって，画面左右のエネルギーバーが増減する。あまりエネルギーが下がると速度が落ちるので，エネルギーと速度の兼ね合いをうまくとってプレイしてほしい。相手が近づくまでエネルギーを温存し，スイッチ１をタイミングよく使うのがコツだ。

プログラムリストを**リスト５，６**に示す。リスト６は SAVEM“ファイル名”，＆H4000，＆H48EF，＆H4000でセーブする。

CLEAR，＆H3FFF ↵
LOADM“リスト６”↵
RUN“リスト５”↵

で走らせる。

おまけ

本誌の７号，４，５，６月号に連載された，「マシン語ゲームの作り方」での「エイリアンブロックくずし」にパッチを当てたので，**リスト７**に紹介する。

基本的には**リスト４**の JOYIN 以下を使用した。これを本誌に掲載されたリストの最後に追加し，プログラム内では少し変更を加え，NOPによりバイト数の調整を行った。FM-7,8 用では，プログラム中のキー

ルーチンに余裕があったのでこのようにし，11では，サブルーチンコールの形ですべて後ろにもっていった。

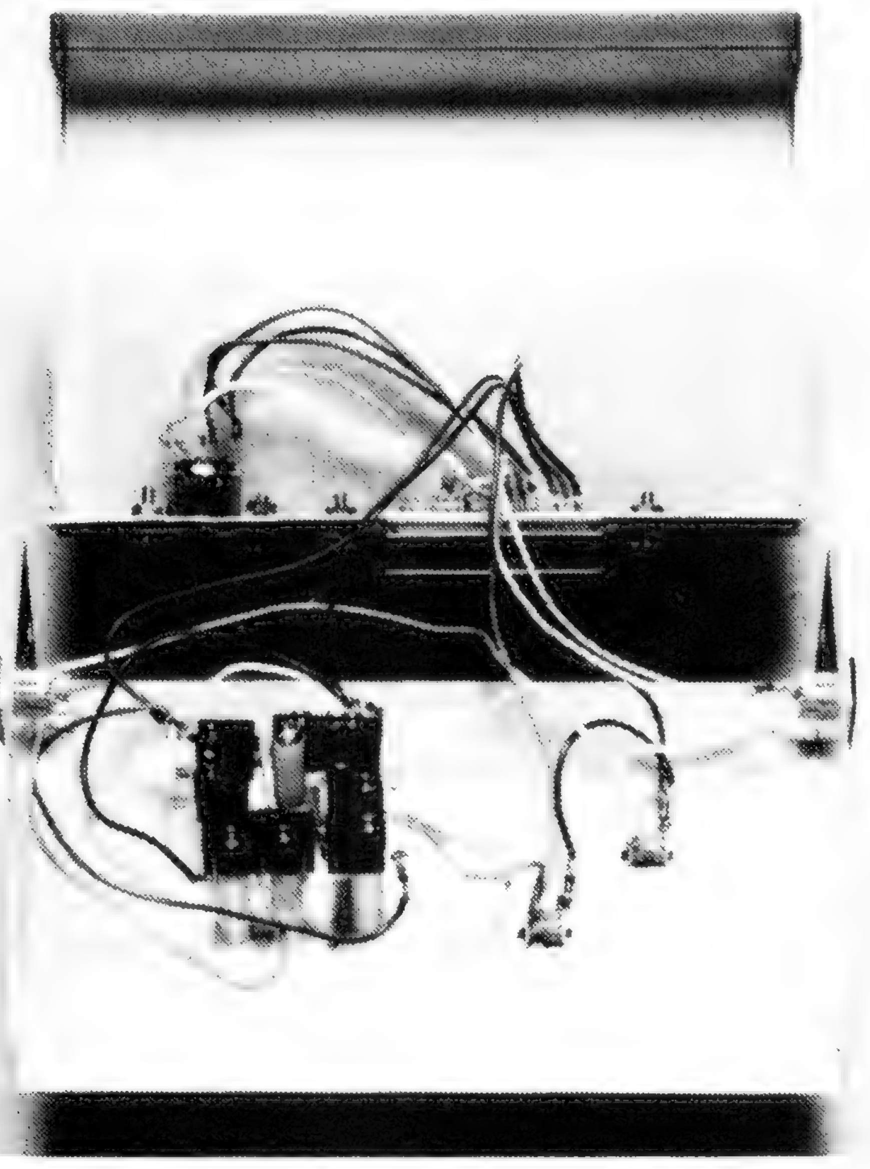
実際にパッチ部分のアセンブラソースを**リスト8**に挙げる。**リスト4**を流用したのでキー入力はずいぶん簡略化されたのがおわかりいただけると思う。また，スイッチ1,2 とジョイスティックのどちらでも動くように工夫したつもりである。ただし，処理には順番があるので，スイッチとジョイスティックの両方を作動させた場合，スイッチが優先されるのは，ご覧のとおりである。**リスト4**とともに，参考にしてほしい。

おわりに

自分の作ったゲームや手持ちのソフトに，このジョイスティックをつなげて，リアルタイム入力を楽しんでいただければ幸いである。なお，本誌にリアルタイムゲームなどの投稿をされる方も，できるだけこのルーチンをつけて，キーボード，ジョイスティックどちらでも動くようにしていただければ，と思っている。

注：㈱アスキーより販売されている「クイックショット」とは，回路・ソフトともに互換性がありませんので注意してください

写真3 ケース内部



```
50B7 CC   0000             LDD   #0
50BA 1083 00C8        CGB02  CMPD  #200
50BE 2B   03   50C3        BMI   CGB03
50C0 CC   00C6             LDD   #198
50C3 1E   01          CGB03  EXG   D,X
50C5 9F   13               STX   LY2
50C7 16   FF6E 5038        LBRA  LOOP
50CA 25   17   50E3 CGCOL  BCS   EXIT     IF SW1=ON and SW2=ON THEN EXIT
50CC 0D   18               TST   TRIG
50CE 26   93   5063        BNE   MOVE
50D0 34   06               PSHS  D
50D2 96   16               LDA   COL
50D4 4C                    INCA
50D5 81   08               CMPA  #8
50D7 2B   02   50DB        BMI   CGC00
50D9 86   01               LDA   #1
50DB 97   16          CGC00  STA   COL
50DD 97   18               STA   TRIG
50DF 35   06               PULS  D
50E1 20   80   5063        BRA   MOVE
50E3 35   FF          EXIT   PULS  CC,D,DP,X,Y,U,PC
                      *
50E5 34   10          JOYIN  PSHS  X        a=0 or 1 -> a=deltaX b=deltaY
50E7 C6   EF               LDB   #%11101111 if a=1 then higher
50E9 30   8C 3E            LEAX  <JOYTAB,PCR
50EC 84   01               ANDA  #1
50EE 26   02   50F2        BNE   JOY00
50F0 C6   FE               LDB   #%11111110 if a=0 then lower
50F2 8D   24   5118 JOY00  BSR   JOYS01   get 2 bit
50F4 34   02               PSHS  A
50F6 1A   01               ORCC  #1
50F8 59                    ROLB
50F9 8D   1D   5118        BSR   JOYS01   get 2 bit
50FB 48                    LSLA
50FC 48                    LSLA
50FD AA   E0               ORA   ,S+      joystick data
50FF 48                    LSLA
5100 AE   86               LDX   A,X      dir->tenkey matrix
5102 34   10               PSHS  X
5104 1A   01               ORCC  #1
5106 59                    ROLB
5107 8D   0F   5118        BSR   JOYS01   get 2 bit
5109 34   02               PSHS  A
510B 1A   01               ORCC  #1
510D 59                    ROLB
510E 8D   08   5118        BSR   JOYS01   get 2 bit
5110 48                    LSLA
5111 48                    LSLA
5112 AA   E0               ORA   ,S+      switch data
5114 1F   8A               TFR   A,CC
5116 35   96               PULS  D,X,PC
5118 F7   FD01        JOYS01 STB   LPTOUT
511B B6   FD02             LDA   LPTIN
511E 43                    COMA
511F 46                    RORA
5120 46                    RORA
5121 34   01               PSHS  CC
5123 46                    RORA
5124 35   01               PULS  CC
5126 49                    ROLA
5127 84   03               ANDA  #%00000011
5129 39                    RTS
                      *
512A      0000        JOYTAB FDB   0
512C      00FF             FDB   $00FF    UP
512E      0001             FDB   $0001    DOWN
5130      0000             FDB   0
5132      0200             FDB   $0200    RIGHT
5134      02FF             FDB   $02FF    R.U.
5136      0201             FDB   $0201    R.D.
5138      0000             FDB   0
513A      FE00             FDB   $FE00    LEFT
513C      FEFF             FDB   $FEFF    L.U.
513E      FE01             FDB   $FE01    L.D.
5140      0000             FDB   0
5142      0000             FDB   0
5144      0000             FDB   0
5146      0000             FDB   0
5148      0000             FDB   0
          5000             END   ENTRY
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

PROGRAM BEGIN ADDR=4F00
PROGRAM END   ADDR=5149
PROGRAM ENTRY ADDR=5000
```

リスト5

●FM-7, 77用

```
100 CLEAR 1000,&H3EFF:DEFUSR0=&H4000
110 DEFINT A-Z
120 FOR A=0 TO 7:COLOR=(A,A):NEXT A
130 CLS:A=USR0(0)
140 B=2-A:A=5+A
150 FOR C=15 TO 1 STEP -1
160 COLOR=(A,0):COLOR=(B,7)
170 PLAY 'E64':FOR D=1 TO C:NEXT D
180 COLOR=(A,7):COLOR=(B,0)
190 PLAY 'G64':FOR D=1 TO C:NEXT D
200 NEXT C:COLOR=(A,0):COLOR=(B,0):LINE(220,169)-(420,191),PSET,0,BF
210 X=225:Y=175:C=4:D=7:X$='Push Space Bar to Replay':GOSUB 230
220 IF INKEY$<>' ' THEN 210 ELSE COLOR=(A,A):COLOR=(B,B):GOTO 130
230 SYMBOL(X+1,Y+1),X$,1,1,C:SYMBOL(X+2,Y+1),X$,1,1,C:SYMBOL(X,Y),X$,1,1,D:RETUR
N
```

●FM-11用追加行

```
50 SCREEN 4,0,0,7,7
```

●FM-8用変更行

```
120 '
140 IF A=0 THEN W$='RIGHT' ELSE W$='LEFT'
160 '
170 BEEP 1:FOR D=1 TO C:NEXT D
180 '
190 BEEP 0:FOR D=1 TO C:NEXT D
200 NEXT C:LINE(220,159)-(420,191),PSET,0,BF
205 X=230:Y=165:C=1+A:D=6-A:X$='*****  '+W$+' win *****':GOSUB 230
220 IF INKEY$<>' ' THEN 210 ELSE 130
```

リスト6

```
ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F =S
4000 34 7E 86 3F 1F 8B 31 8D 06 FA 17 05 79 31 8D 06 38
4010 FE 17 05 72 17 03 44 31 8D 06 FF 17 05 68 CC 00 FD
4020 64 DD 16 DD 18 DD 2E C6 01 DD 1E DD 20 DD 22 DD F2
4030 38 DD 3C C6 03 DD 34 DD 36 C6 31 DD 24 DD 3A C6 13
4040 02 DD 26 C6 05 DD 28 C6 06 DD 3E CC 01 F4 DD 2C 86
4050 31 8D 06 D1 17 05 2F 31 8D 06 D5 17 05 28 31 8D 7B
4060 06 D9 17 05 21 31 8D 06 DD 17 05 1A CC 00 00 DD 9C
4070 00 DD 02 DC 26 DD 10 DC 28 DD 0A 30 8D 06 DD 17 70
4080 05 2C CC 02 45 DD 00 DC 3C DD 10 DC 3E DD 0A 30 57
4090 8D 06 D1 17 05 18 CC 00 00 DD 46 DC 46 C3 00 01 6D
40A0 10 83 00 09 26 03 CC 00 01 DD 46 10 93 22 10 23 AD
40B0 00 06 17 08 27 16 01 3E 86 00 17 05 F3 DD 48 C4 1F
40C0 01 27 1A DC 24 10 83 00 AA 24 12 DC 22 C3 00 01 77
40D0 10 83 00 08 2D 03 CC 00 08 DD 22 20 0C DC 22 83 4B
40E0 00 01 26 03 CC 00 01 DD 22 DC 24 D3 22 83 00 04 72
40F0 10 83 00 08 2A 03 CC 00 08 10 83 00 BE 2D 03 CC E9
---------------------------------------------------------
SUM  CA 58 16 E5 92 5C 80 31 01 FE 15 9F 39 63 27 C2 F4

ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F =S
4100 00 BE DD 2A DC 2A 93 24 23 2A CC 00 00 ED 8D 05 1A
4110 96 ED 8D 05 96 DC 24 ED 8D 05 8E DC 2A 83 00 01 42
4120 ED 8D 05 89 CC 00 00 ED 8D 05 7A A7 8D 05 80 17 9D
4130 04 50 20 29 CC 00 00 ED 8D 05 6C ED 8D 05 6C DC 1B
4140 24 ED 8D 05 64 DC 2A ED 8D 05 62 DC 26 E7 8D 05 69
4150 54 4F A7 8D 05 50 A7 8D 05 55 17 04 25 DC 48 81 9F
4160 FF 27 2C D6 21 58 50 DB 48 CB 08 C4 07 27 20 C1 BA
4170 05 2A 10 C1 04 27 18 DC 20 C3 00 05 C4 03 4F DD FA
4180 48 20 10 DC 20 C3 00 03 C4 03 4F DD 48 20 04 DC 75
4190 20 DD 48 DC 48 DD 1E 17 03 D5 D3 16 DD 1A DC 1E 2D
41A0 17 03 D7 D3 18 DD 1C DC 1A ED 8D 04 AE ED 8D 04 75
41B0 AE DC 1C ED 8D 04 A6 ED 8D 04 A6 17 04 56 4D 10 BC
41C0 26 01 84 DC 1E DD 08 DC 20 DD 0A DC 16 DD 00 DC 18
41D0 18 DD 02 DC 1A DD 04 DC 1C DD 06 DC 26 DD 4A DC AE
41E0 28 DD 12 17 01 BC DC 1A DD 16 DC 1C DD 18 DC 1E BB
41F0 DD 20 DC 2A DD 24 DC 46 10 93 38 10 23 00 06 17 51
---------------------------------------------------------
SUM  73 CC BE 7B BB CC 94 17 5B 4D 3A 0B 6D B6 A3 18 75

ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F =S
4200 06 DA 16 01 3C 86 01 17 04 A6 DD 48 C4 01 27 1A A6
4210 DC 3A 10 83 00 AA 24 12 DC 38 C3 00 01 10 83 00 F4
4220 08 2D 03 CC 00 08 DD 38 20 0C DC 38 83 00 01 26 0B
4230 03 CC 00 01 DD 38 DC 3A D3 38 83 00 04 10 83 00 20
4240 08 2A 03 CC 00 08 10 83 00 BE 2D 03 CC 00 BE DD F1
4250 40 DC 40 93 3A 23 2A CC 02 7F ED 8D 04 49 ED 8D 04
4260 04 49 DC 3A ED 8D 04 41 DC 40 83 00 01 ED 8D 04 40
4270 3C CC 00 00 ED 8D 04 2D A7 8D 04 33 17 03 03 20 5B
4280 29 CC 02 7F ED 8D 04 1F ED 8D 04 1F DC 3A ED 8D 40
4290 04 17 DC 40 ED 8D 04 15 DC 3E E7 8D 04 07 4F A7 59
42A0 8D 04 03 A7 8D 04 08 17 02 D8 DC 48 81 FF 27 2C BC
42B0 D6 37 58 50 DB 48 CB 08 C4 07 27 20 C1 05 2A 10 BD
42C0 C1 04 27 18 DC 36 C3 00 05 C4 03 4F DD 48 20 10 49
42D0 DC 36 C3 00 03 C4 03 4F DD 48 20 04 DC 36 DD 48 6E
42E0 DC 48 DD 34 17 02 88 D3 2C DD 30 DC 34 17 02 8A 95
42F0 D3 2E DD 32 DC 30 ED 8D 03 61 ED 8D 03 61 DC 32 E6
---------------------------------------------------------
SUM  51 F6 25 1E 41 47 36 5A F8 20 CE 13 46 95 D1 52 99

ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F =S
4300 ED 8D 03 59 ED 8D 03 59 17 03 09 4D 26 43 DC 34 95
4310 DD 08 DC 36 DD 0A DC 2C DD 00 DC 2E DD 02 DC 30 B8
4320 DD 04 DC 32 DD 06 DC 3C DD 4A DC 3E DD 12 17 00 31
4330 71 DC 30 DD 2C DC 32 DD 2E DC 34 DD 36 DC 40 DD BB
4340 3A 17 05 8D 16 FD 54 35 7E 34 06 86 00 A7 03 35 9C
4350 86 35 7E 34 06 86 01 A7 03 35 86 30 8D 04 57 CC 43
4360 00 32 DD 00 CC 00 50 DD 02 CC 00 02 DD 0A 4F A7 B5
4370 8D 03 3C E6 80 DD 08 E6 80 10 27 01 77 DD 42 DD 28
4380 44 DC 08 17 01 E9 D3 00 DD 04 DC 08 17 01 EB D3 97
4390 02 DD 06 CC 00 02 DD 4A CC 00 07 DD 12 8D 03 16 42
43A0 01 2C DC 08 C4 01 10 27 00 67 DC 0A 26 33 DC 08 97
43B0 17 01 BC 57 D3 00 ED 8D 02 ED DC 02 C3 00 01 ED F6
43C0 8D 02 E6 DC 04 C3 00 01 ED 8D 02 DF DC 06 C3 00 19
43D0 01 ED 8D 02 D8 DC 4A E7 8D 02 CA 17 01 A4 16 00 8D
43E0 9F DC 00 C3 00 02 ED 8D 02 BD DC 02 C3 00 01 ED 08
43F0 8D 02 B6 DC 04 C3 00 02 ED 8D 02 AF DC 06 C3 00 BA
---------------------------------------------------------
SUM  7D A9 56 04 B3 29 7E B2 16 9F ED E7 85 36 62 91 C3

ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F =S
4400 01 ED 8D 02 A8 DC 4A E7 8D 02 9A 17 01 74 16 00 FD
4410 6F DC 0A C1 03 26 36 DC 00 C3 00 01 ED 8D 02 87 18
4420 DC 08 17 01 55 57 D3 02 ED 8D 02 7D DC 04 C3 00 19
4430 02 ED 8D 02 76 DC 06 ED 8D 02 72 DC 4A E7 8D 02 60
4440 64 86 02 A7 8D 02 68 17 01 38 16 00 33 DC 00 C3 C2
4450 00 01 ED 8D 02 51 DC 02 C3 00 01 ED 8D 02 4A DC 12
4460 04 C3 00 02 ED 8D 02 43 DC 06 C3 00 01 ED 8D 02 AA
4470 3C DC 4A E7 8D 02 2E 86 02 A7 8D 02 32 17 01 02 10
4480 DC 00 ED 8D 02 21 DC 02 ED 8D 02 1D DC 04 ED 8D 4A
4490 02 19 DC 06 ED 8D 02 15 DC 12 E7 8D 02 07 4F A7 EF
44A0 8D 02 0C 17 00 DC DC 08 C4 01 26 21 DC 00 83 00 DD
44B0 01 ED 8D 01 F2 DC 02 ED 8D 01 EE DC 04 83 00 01 19
44C0 ED 8D 01 E7 DC 06 ED 8D 01 E3 17 00 B5 39 DC 04 87
44D0 DD 00 DC 06 DD 02 DC 08 DD 0A DC 44 83 00 01 DD EA
44E0 44 10 26 FE 9C 17 01 99 5D 10 27 FE 81 81 20 10 89
44F0 26 FE 7B 39 CC 01 C2 DD 00 CC 00 64 DD 02 CC 00 1F
---------------------------------------------------------
SUM  92 87 54 B2 81 9D 15 AB FE A3 8C AD 5B 18 C8 52 64
```

```
ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F =S
4500 02 DD 10 CC 00 07 DD 0A 30 8D 02 60 17 00 9F CC 4A
4510 00 6E DD 02 CC 01 BE DD 00 30 8D 02 5B 17 00 8E 74
4520 CC 00 78 DD 02 CC 01 C2 DD 00 30 8D 02 52 17 00 B7
4530 7D CC 00 82 DD 02 CC 01 BE DD 00 30 8D 02 4E 17 36
4540 00 6C CC 00 8C DD 02 CC 01 C2 DD 00 30 8D 02 46 14
4550 17 00 5B CC 01 04 DD 00 CC 00 B4 DD 02 30 8D 02 3E
4560 46 17 00 4A 17 01 1A 5D 27 E9 81 20 26 E5 39 34 5F
4570 10 30 8D 03 21 58 EC 8B 35 90 34 10 30 8D 03 1E A7
4580 20 F3 34 16 20 0F 34 16 30 8D 01 19 C6 0B A6 A0 C4
4590 A7 80 5A 26 F9 30 8D 01 09 AF 8D 00 FF CC 00 0E 7C
45A0 ED 8D 00 FA 30 8D 00 F2 AD 9F FB FA 35 96 31 8D ED
45B0 03 00 A6 80 A7 A0 E6 80 E7 A0 4A 26 F9 31 8D 02 86
45C0 ED DC 00 C3 00 01 ED A1 DC 02 C3 00 01 ED A4 DC 2A
45D0 10 E7 8D 02 D4 8D 29 EC 8D 02 D3 C3 00 01 ED 8D 9C
45E0 02 CC 8D 1C EC 8D 02 C6 83 00 02 ED 8D 02 BF EC 64
45F0 8D 02 BD 83 00 01 ED 8D 02 B6 DC 0A E7 8D 02 A9 07
-------------------------------------------------------
SUM  FB 5B 24 60 20 98 F9 C7 AF 0A 4C 1F F1 B5 85 46 E7
```

```
ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F =S
4600 34 16 30 8D 02 A0 AF 8D 00 92 CC 00 2B ED 8D 00 E8
4610 8D 16 FF 90 34 10 86 1D A7 8D 00 3E 30 8D 00 38 80
4620 AF 8D 00 2E CC 00 0B ED 8D 00 29 CC 00 14 ED 8D 3E
4630 00 24 30 8D 00 1A 86 11 A7 8D 00 14 AD 9F FB FA 1B
4640 A6 8D 00 18 AA 8D 00 15 AA 8D 00 12 84 80 35 90 A9
4650 11 00 00 00 00 00 00 00 00 00 1D 00 00 00 00 00 2E
4660 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4670 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4680 00 34 10 86 15 A7 8C C8 30 8C CB AF 8C C4 30 8C 1C
4690 BF AD 9F FB FA EC 8C BE 35 90 10 00 00 00 00 00 0B
46A0 00 00 00 00 15 00 00 00 00 00 00 00 00 00 00 00 15
46B0 34 10 C6 EF 30 8C 3D 84 01 26 02 C6 FE 8D 23 34 47
46C0 02 1A 01 59 8D 1C 48 48 AA E0 A6 86 34 02 1A 01 B6
46D0 59 8D 0F 34 02 1A 01 59 8D 08 48 48 AA E0 1F 89 F6
46E0 35 92 F7 FD 01 B6 FD 02 43 46 46 34 01 46 35 01 F1
46F0 49 84 03 39 FF 00 04 FF 02 01 03 FF 06 07 05 FF 21
-------------------------------------------------------
SUM  F3 18 DE 23 8F 62 65 69 67 AA 26 A6 FB 2D 70 99 D9
```

```
ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F =S
4700 FF FF FF FF 07 00 00 0B 00 08 02 74 00 C7 01 07 5B
4710 00 00 08 00 09 02 77 00 C6 01 00 00 00 0C 00 0A 67
4720 02 73 00 C5 02 02 00 00 00 00 31 00 00 00 C7 00 36
4730 00 00 00 00 00 08 00 00 00 30 00 06 00 02 7F 00 BF
4740 31 02 7F 00 C7 00 00 00 02 7F 00 08 02 7F 00 30 B3
4750 00 00 00 00 0C 00 0A 02 73 00 C5 02 07 50 4C 41 36
4760 59 45 52 31 07 50 4C 41 59 45 52 32 0B 75 73 65 7F
4770 20 50 50 53 54 49 43 4B 07 20 20 20 20 20 55 50 8A
4780 0C 4C 45 46 54 20 2B 20 52 49 47 48 54 08 20 20 68
4790 20 20 44 4F 57 4E 10 53 57 31 20 3A 20 48 69 67 F5
47A0 68 20 53 70 65 65 64 0E 50 75 73 68 20 53 70 61 6B
47B0 63 65 20 42 61 72 01 0A 00 05 01 0A 00 0A 03 0A 2F
47C0 00 0A 01 0A 02 01 03 09 02 08 01 0A 02 0C 03 0A 54
47D0 02 04 01 0F 00 19 01 0A 02 0A 03 09 02 0F 01 0F 73
47E0 00 19 01 0A 02 01 03 09 02 09 01 08 02 01 03 08 55
47F0 02 09 01 0A 02 01 03 0A 02 04 01 0F 00 19 01 0A 60
-------------------------------------------------------
SUM  A6 2A 28 BC B7 06 BA 4A 9C 30 4B F4 CE 1B 5F 54 1C
```

```
ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F =S
4800 02 01 03 09 02 09 01 08 02 01 03 08 02 09 01 0A 47
4810 02 01 03 0A 02 04 01 0F 00 19 01 05 02 01 01 05 4E
4820 02 13 03 05 02 01 03 04 02 04 01 0F 02 01 03 3C 7F
4830 02 23 01 0A 00 18 03 06 00 01 01 0D 02 01 03 06 6C
4840 02 18 01 0A 00 19 01 0A 02 0A 03 09 02 01 01 01 66
4850 02 01 01 01 02 01 01 01 02 01 01 01 02 01 01 01 14
4860 02 01 01 01 02 01 01 01 02 01 01 01 02 01 01 01 14
4870 02 01 01 01 02 05 01 05 00 19 01 0A 02 09 03 09 4D
4880 02 01 01 09 02 0F 01 05 00 19 01 0A 02 0A 03 09 60
4890 02 0F 01 0F 00 00 00 00 00 04 00 00 FF FC FF FE 1D
48A0 00 00 00 02 00 00 00 00 19 00 00 00 01 01 00 00 1D
48B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
48C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
48D0 00 34 10 8E 03 00 30 1F 26 FC 35 90 34 10 8E 04 E1
48E0 00 20 F3 00 00 00 00 00 00 00 00 00 00 00 00 00 13
48F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
-------------------------------------------------------
SUM  14 B7 13 D7 11 55 3D 56 49 5D 42 D8 46 2F 9E 68 E9
```

$48D4・D5番地………ウエイト　0001………FFFF,0000
(小)←――――→(大)

リスト7

●FM-7・77・8用パッチ

```
Add   +0 +1 +2 +3 +4 +5 +6 +7  Sum
5268  B6 59 73 B7 59 74 12 12 :2A
5270  12 12 12 12 12 12 12 12 :90
5278  12 12 12 12 12 4F 17 09 :C9
5280  34 29 0C 25 0E 81 02 27 :46
5288  06 81 FE 27 06 20 19 86 :71
5290  01 20 02 86 FF BB 59 81 :3D

5BB0  02 01 02 01 02 34 10 C6 :12
5BB8  EF 30 8C 3E 84 01 26 02 :96
5BC0  C6 FE 8D 24 34 02 1A 01 :C6
5BC8  59 8D 1D 48 48 AA E0 48 :65
5BD0  AE 86 34 10 1A 01 59 8D :79
5BD8  0F 34 02 1A 01 59 8D 08 :4E
5BE0  48 48 AA E0 1F 8A 35 96 :8E
5BE8  F7 FD 01 B6 FD 02 43 46 :33
5BF0  46 34 01 46 35 01 49 84 :C4
5BF8  03 39 00 00 00 FF 00 01 :3C
5C00  00 00 02 00 02 FF 02 01 :06
5C08  00 00 FE 00 FE FF FE 01 :FA
5C10  00 00 00 00 00 00 00 00 :00
5C18  00 00 00 00 00 00 00 00 :00
```

●FM-11用パッチ

```
Add   +0 +1 +2 +3 +4 +5 +6 +7  Sum
2268  30 12 12 12 12 12 12 12 :AE
2270  12 12 12 12 12 12 12 12 :90
2278  12 12 12 12 12 12 12 16 :94
2280  08 E0 BB 29 3D 81 02 25 :B1

2B60  01 02 4F 8D 19 29 0D 25 :53
2B68  10 81 02 27 07 81 FE 27 :67
2B70  08 16 F7 21 86 01 16 F7 :CA
2B78  09 86 FF 16 F7 04 34 10 :E3
2B80  C6 EF 30 8C 3E 84 01 26 :5A
2B88  02 C6 FE 8D 24 34 02 1A :C7
2B90  01 59 8D 1D 48 48 AA E0 :1E
2B98  48 AE 86 34 10 1A 01 59 :34
2BA0  8D 0F 34 02 1A 01 59 8D :D3
2BA8  08 48 48 AA E0 1F 8A 35 :00
2BB0  96 F7 FD 01 B6 FD 02 43 :83
2BB8  46 46 34 01 46 35 01 49 :86
2BC0  84 03 39 00 00 00 FF 00 :BF
2BC8  01 00 00 02 00 02 FF 02 :06
2BD0  01 00 00 FE 00 FE FF FE :FA
2BD8  01 00 00 00 00 00 00 00 :01
2BE0  00 00 00 00 00 00 00 00 :00
```

リスト8

```
03020                                   * JOYSTICK PATCH
03030   527D 4F                                 CLRA
03040   527E 17      0934 5BB5                  LBSR    JOYIN
03050   5281 29      0C   528F                  BVS     PDL1       SW1
03060   5283 25      0E   5293                  BCS     PDL2       SW2
03070   5285 81      02                         CMPA    #2
03080   5287 27      06   528F                  BEQ     PDL1       RIGHT
03090   5289 81      FE                         CMPA    #-2
03100   528B 27      06   5293                  BEQ     PDL2       LEFT
03110   528D 20      19   52A8                  BRA     PDL3       DON'T MOVE
03120   528F 86      01              PDL1       LDA     #1
03130   5291 20      02   5295                  BRA     PDL20
03140   5293 86      FF              PDL2       LDA     #-1
03150   5295 BB      5981            PDL20      ADDA    PXX
```

for FM-7, 8, 11

効率的なビジネスソフト作成のために

ビジネス用BASICプログラムジェネレータ

(6)ソートプログラムジェネレータ

高木　征蔵

1 ビジネス処理とソートプログラム

本題に入る前に簡単に前回のマージプログラムジェネレータの章で述べた要点をまとめて，ビジネス処理におけるソートプログラムについて復習してみましょう。

データをある項目の小さい順，あるいは大きい順に並べることをソートといい，ビジネス処理の場合，これは通常ファイルに記録されているデータレコードの並べかえになります。1つのデータレコードのなかにはいろいろな項目が含まれており，ソートとは，このなかから選ばれた項目についてそれの小さい順(昇順)，または大きい順（降順）にデータレコードを並べかえることを意味します。ソートキー項目は一度に2つ以上でもよく，この場合はそれぞれについて昇順か降順かを指定します。

ファイルをソートする場合，一般にはそのファイルに含まれるすべてのデータをメモリ内部に読み込むことは不可能です。そこで，メモリ内部に読み込める範囲の件数のデータレコードを読み込み，メモリ内部で指定されたキー項目の優先順序にしたがった並べかえをした後，これらの1グループのデータレコードを，1つの小さなファイルとしてひとまず外部へ書き出します。これをくり返すことによってソート済みの小規模ファイルが多数できますので，これらをマージしていけば最終的にソートされた1つのファイルが得られます。

以上がファイルソートの概要です。前回のマージプログラムジェネレータの章で，ソート，あるいはマージのキー項目と条件が指定されたときの2つのデータレコードの優先順位の判定と，複数のファイルのマージ処理について説明しました。これらはいずれも，ソートプログラムで重要な役割を果たします。詳しい内容は前号を参照していただくことにして，今回はこの知識を前提にして進みます。

ソートプログラムは次の2つの要素

①メモリ内部でのデータの並べかえ

②ソート済みの多数の小規模ファイルのマージ

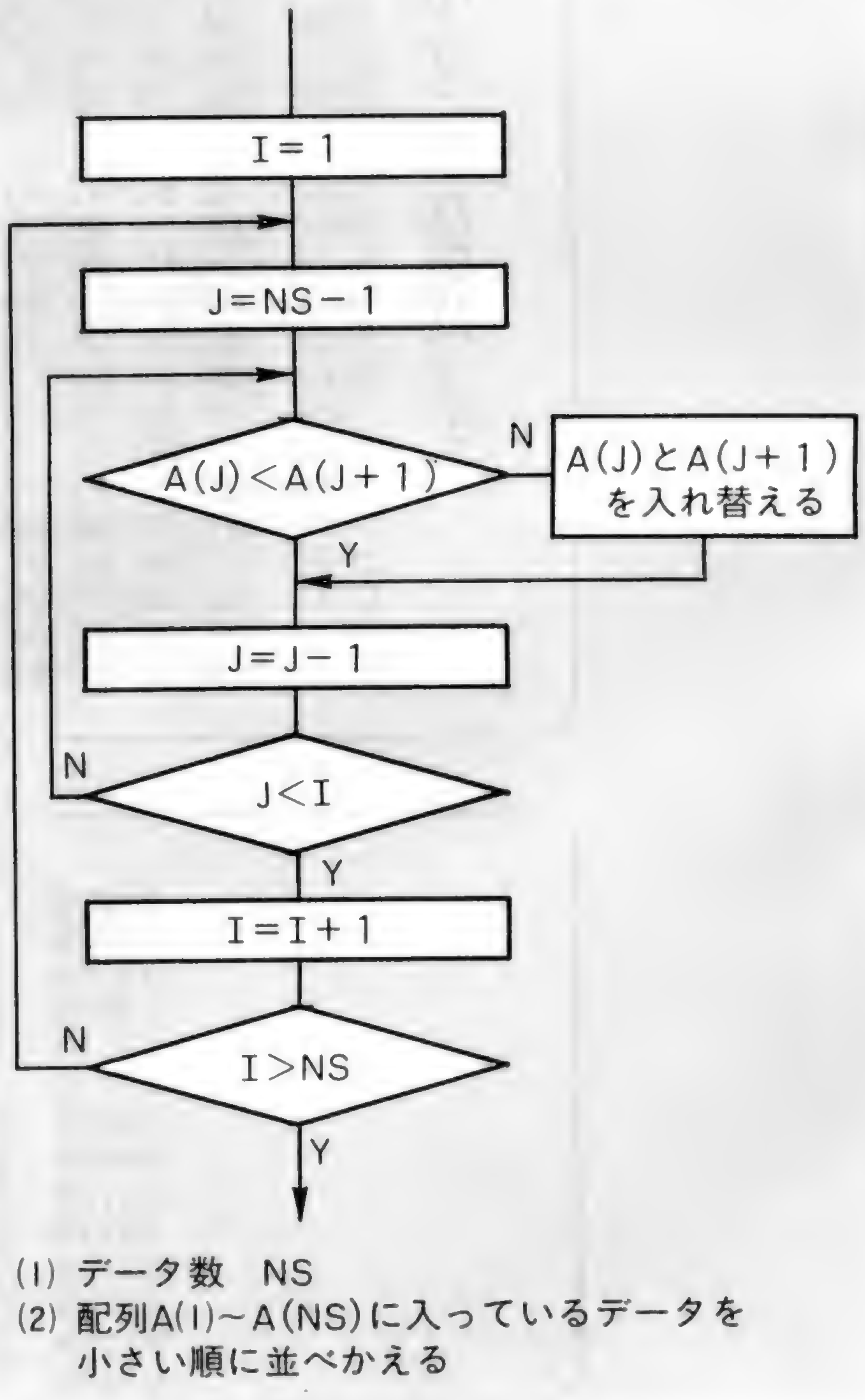
から構成されます。マージ処理については前回詳しく説明しましたので，ソートプログラムジェネレータ作成の主要課題はメモリ内部でのデータの並べかえと，多数の小規模ファイルのマージの手順の2つに絞られます。

本シリーズではファイル構成として，S（通常のシーケンシャルファイル），R（通常のランダムファイル），SR（ランダムファイルと同じデータの詰め方をしたシーケンシャルファイル）の3種類を扱っています。ソート処理ではプログラムの性能を高めるためにSファイルは除外し，RファイルとSRファイルのみを対象にします。Sファイルはいったん SR ファイルに変換してからソートし，ソート終了後，再びSファイルに変換します。この変換処理をするプログラムのジェネレータ“SSRGEN”については前回で紹介しました。

2 メモリ内部でのデータの並べかえ

ソートの基本形はメモリ内部に読み込まれたデータを並べかえることです。並べかえの方法は，実行時間の短縮を考慮して多くの形が考えられていますが，ここではもっとも単純な方式である逐次交換型を採用します。この方式のフローチャートを**図1**に，プログラム例を**リスト1**に示します。いきなりフローチャートとプログラムが出てきたのではわかりにくいでしょうから，到着順に並んだ人の行列を背の低い順に並べかえるという例で説明しましょう。まず一番後ろと，後ろから2番目の人を比べ，順序が正しければそのままにし，逆なら入れ替えます。次に後ろから2番目と後ろから3番目の人について同様に処理します。これを順次くり返していき，一番前の人の所までやります。これによって一番背の低い人は一番前にきました。そこで次はこの人を除いてその1つ後ろの人の所までまったく同様のことを行います。これによって2番目に背の低い人が前から2番目にきました。以下同様のことをくり返して後ろから2番目の人まで確定すれば，行列は背の

図1　逐次交換型ソートのフローチャート



(1) データ数　NS
(2) 配列A(1)～A(NS)に入っているデータを小さい順に並べかえる

低い順に並べかえられたことになります。この方法はデータ件数の2乗に比例する回数の処理が必要なため，件数が増えると長い時間がかかるのが欠点ですが，いたずらに話をむずかしくしないために，ここではこの方式を採用することにします。メモリ内部に読み込むデータの件数を変数にとってもよいのですが，話をいたずらに複雑にしないために，処理時間とメモリの大きさの関係でここでは一定値40を採用します。これは最大256バイトのデータレコードの長さをもつファイルが5インチミニフロッピに入っている場合，FM-7,8でのソート時間をほぼ最小にする値です。

ソート処理に要する時間は次の2つ

(1)メモリ内部での並べかえに要する時間

(2)ソート済みの小規模ファイルを読み書きしてマージする時間

に大きく分けられます。メモリ内部に読み込むデータの件数を少なくすれば，(1)は小さくなりますが，(2)は大きくなります。逆にメモリに読み込むデータ件数を多くすると，(1)は大きくなり，(2)は小さくなります。FM-7，8でいくつかのファイルについてこの両方の時間を測定し，結果として得られたのが上記の40という値です。

ちょっと話が横にそれましたが，**図1**のフローチャートは理解されたことと思います。そこで，メモリ内部でのデータの並べかえの実感をつかむために，**リスト1**のプログラムを打ち込んで実行してみましょう。最初に乱数発生関数によって作り出された元データの配列が表示され，次にソート開始時刻，続いてソート終了時刻が表示され，最後に並べかえられたデータの配列が表示されます。データ件数NSをいろいろ変えて並べかえに要する時間を測定してみると，データ件数の2乗で長くなっていくことがわかるでしょう。

3 ファイルソートの実際

それではファイルソートの実際的な手順についてまとめてみましょう。具体的なプログラムは意識しないで基本的な考え方を整理します。

(1)入力ファイルをOPENする。

(2)入力ファイルからデータを40件だけメモリ内部に読み込み，指定された条件で並べかえた後，これを1つの独立したファイルとして書き出す。

(3)入力ファイルのデータがなくなるまで(2)をくり返す。ただし，入力ファイルから読み込むデータが40件に満たないで終わってしまった場合は，そこまでのデータについて実行する。

(4)このようにして作られたソート済みデータのファイルを次々とマージして，1つのソート済みデータのファイルを作成する。

(5)このマージのやり方としては，2つずつマージしていってファイルを倍々に長くしていく方法と，全部のファイルを一度にマージしてしまう方法とがある。

以上が一般的なファイルソートの手順ですが，これをシーケンシャルファイルとランダムファイルに分けてもう少し詳しく考えてみましょう。

シーケンシャルファイルの場合は，1つのファイルで読み書きはできませんので，メモリ内部で並べかえられたデータは新しいファイルとして書き出さねばなりません。一般にはこのような小規模ファイルが多数作り出されますが，この段階が終わった所で元の入力ファイルをKILLします。次にこれらの小規模ファイルをマージして作り出されるファイルを，このKILLされた入力ファイルの書かれていた領域に書き込むことができます。すなわち，小規模ファイルのマージが終わった時点で古いファイルをKILLするようにすれば，フロッピディスク領域の有効活用がはかれます。このような工夫をすることにより，一度にソートできるデータの件数を大きくすることができます。

ランダムファイルの場合は，1つのファイルで読み書きが同時にできますし，またファイル内のどの位置からもデータを読み書きすることができます。したがって，入力ファイルを保存する必要がなければ，メモリ内部に読み込んで並べかえられたデータは，シーケンシャルファイルのように別のファイルとして書き出さないで，そのま

©1984 S. TAKAGI

リスト1●逐次交換型ソートのプログラム例

```
10 '    "SORTREI"    Sort Sample
20 DIM A(1000)
30 INPUT "ﾃﾞｰﾀ ﾉ ｶｽﾞ = ",NS
40 FOR I=1 TO NS:A(I)=INT(1000*RND(I)):NEXT I
50 FOR I=1 TO NS:PRINT A(I):NEXT I
60 PRINT "Sort Start at "+TIME$:T1=TIME
70 FOR I=1 TO NS
80 FOR J=NS-1 TO I STEP -1
90 IF A(J)>A(J+1) THEN SWAP A(J),A(J+1)
100 NEXT J
110 NEXT I
120 PRINT "Sort End   at "+TIME$:T2=TIME
130 FOR I=1 TO NS:PRINT A(I):NEXT I
140 PRINT "Sort Time  = "+STR$(T2-T1)+" 秒"
150 END
```

リスト1のBASICチェックサム

```
LINE +0 +1 +2 +3 +4 +5 +6 +7 SUM
  10 00 2D C2 9B 46 21 E2 F3 :C6
  90 8B 89 88 6B 46 94 D7 00 :B8
---------------------------------
     8B B6 4A 06 8C B5 B9 F3 /7E
```

注）本記事のすべてのプログラムはFM-11BSでも動きます（F-BASIC86を起動のこと）。

ま元のファイルの同じ領域に書き込むことができます。プログラムのなかでこの区切りのレコード位置を覚えておけば，1つのファイルのなかに見かけ上多数のファイルがある形となり，マージもこの仮想的なファイルの単位で行い，最終的な出力ファイルだけ別の領域を割り当てればよいということになります。しかも，仮想的なファイルは多数あっても，プログラム上のファイルは1つですから，これらの仮想的なファイルからデータを1つずつとり出して一度にマージすることが可能です。このマージの出力ファイルの領域だけ新たにとればよいわけです。このような工夫で，やはり一度にソートできるデータの件数を大きくすることができます。

それではこのような工夫によって，ソートできるファイルの最大のサイズはどの程度かを考えてみましょう。フロッピディスクが1台か2台か，元ファイルを保存するか否かによって変わってきますが，シーケンシャルファイル，ランダムファイルともに次のとおりです（5インチ両面倍密度の場合）。

- フロッピディスク1台，元ファイル保存の場合……フロッピ1枚の約1/3(≦50クラスタ)
- フロッピディスク1台，元ファイルを保存しない場合……フロッピ1枚の約1/2（≦70クラスタ）
- フロッピディスク2台の場合……フロッピ約1枚分（≦150クラスタ）

理由は図2を見てください。ランダムファイル，シーケンシャルファイルのいずれにしても，マージの入力ファイルと出力ファイルのためにほぼ同程度のワークエリアが必要です。したがって，フロッピディスクが1台の場合，元ファイルを保存する必要があるならば，ワークエリアはその残りの領域を使わねばならないため，ソート可能なファイルの最大の大きさはフロッピ1枚分の約1/3になります。元ファイルを保存する必要がなければ，その領域をワークエリアに使えるので，フロッピ1枚分の約1/2相当の大きさまで可能となります。フロッピディスクが2台ある場合は，入力ファイルの処理が終わった時点で新しいフロッピと取り換えてしまいます。そうすれば1枚分がそっくりワークエリアとして使えますので，ソート可能なファイルの最大サイズはフロッピ約1枚分となります。

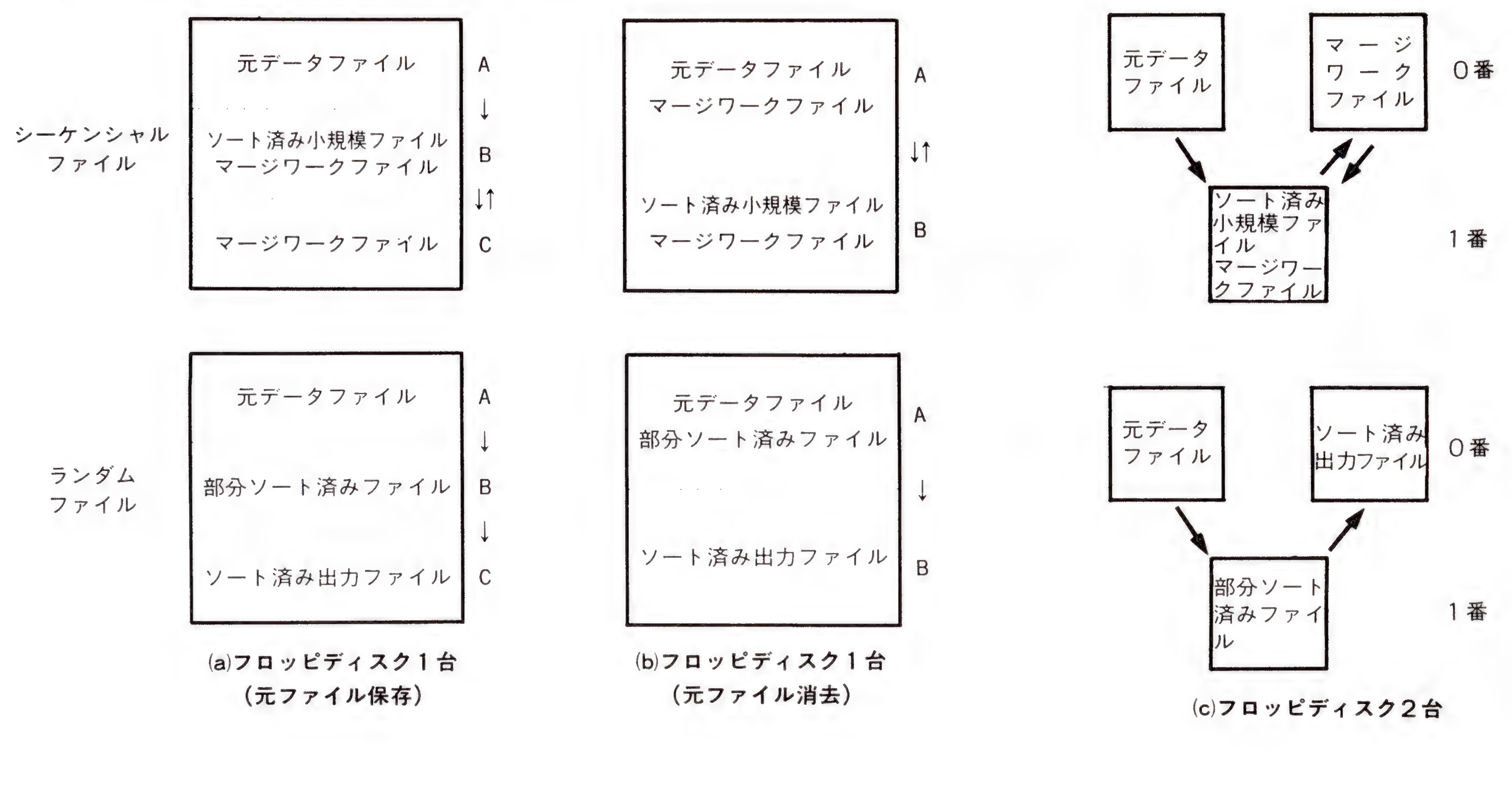
4 シーケンシャルファイルのソートの手順

ソートの手順についての基本的な考え方の整理が終わったところで，具体的なプログラムを意識した，より詳しいシーケンシャルファイルのソートに関する流れをフローチャートに整理してみることにします。ここで扱うシーケンシャルファイルは前に説明したようにSRファイルに限定します。図3のフローチャートにしたがって説明しましょう。

(1) ソート済み小規模ファイルの作成

メモリ内部に読み込んで並べかえるデータの件数をBLとします。BLの値はシーケンシャルファイルのデータレコードの長さ，FMシリーズコンピュータのメモリサ

図2 ソートの際のファイルのフロッピ領域割り当て



イズによって異なりますが，前に述べたように，ここでは40という値を採用します。

まず元ファイルをOPENし，データレコードをBL個だけメモリに読み込み，指定されたソートキーの順序に並べかえます。これは一般にコアソートと呼ばれているもので，2章で説明した方法で実行できます。並べかえの終わったデータは，ファイル名としてSWKA-XXをつけてフロッピに書き出します。ここでSWKはSort Workの意でとったもので，Aの所は以下の進行過程でB, C, D,……と変わっていきます。XXは1から99までの数字を当てます。最初のBL個のソート済みデータのファイル名はSWKA-1，次のBL個はSWKA-2, ……という具合です。最後は一般にはBL個に満たない場合が多いのですが，これはこれだけで並べかえて1つのファイルに書き出します。このようにして作られたソート済みの小規模ファイルの数をNFとします。

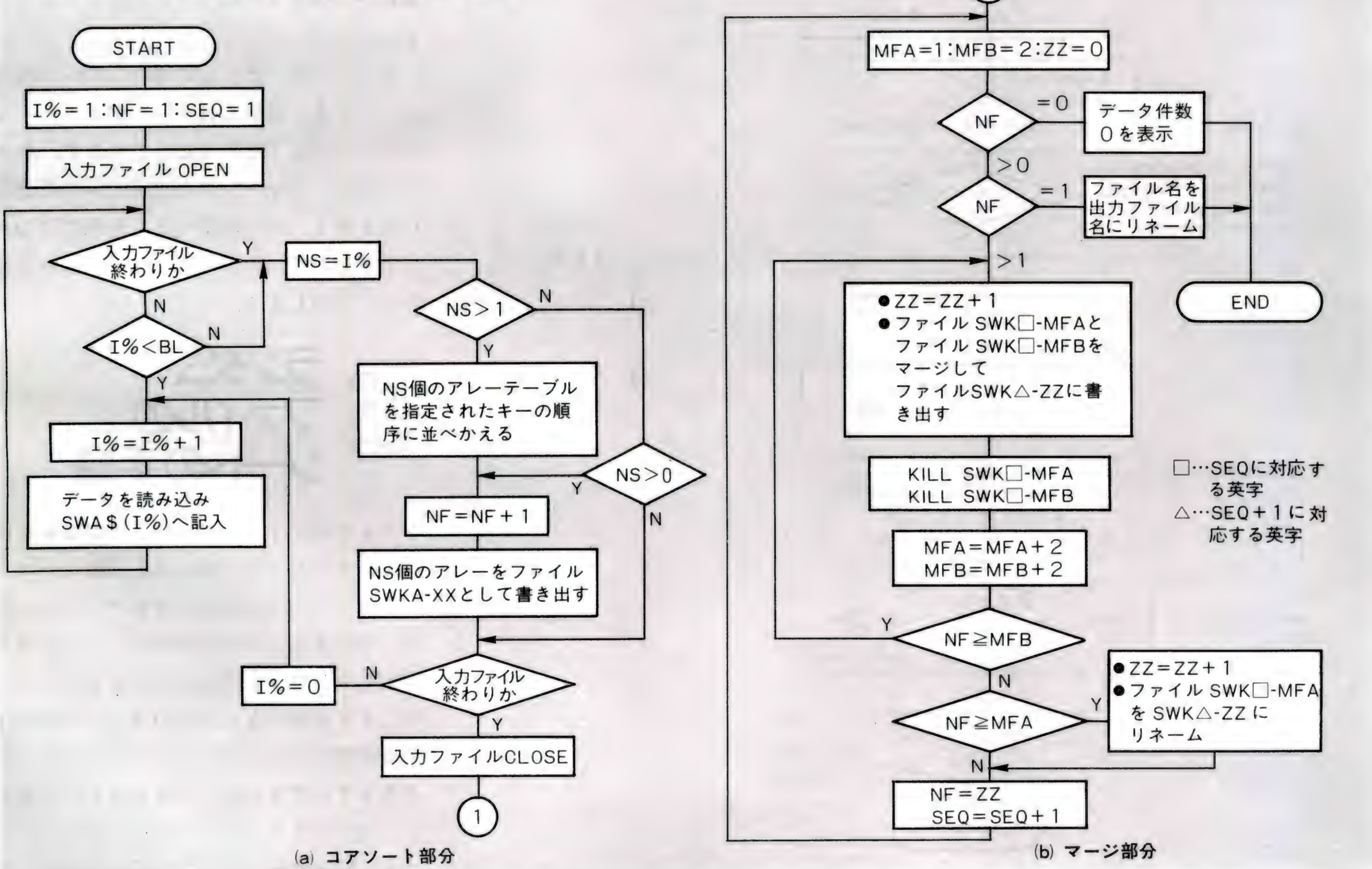
以上の手順は**図3(a)**のソート部分に相当します。図ではSRファイルのデータレコードをメモリ内に格納する領域として，SWA$(50)を使っています。I%はコアソートのために配列SWA$のなかに入っているデータ数のカウンタです。コアソート対象のデータ数は一般にはBLですが，最後はBLに満たない場合が多いので，プログラム上はこのデータ数をNSとします。SEQはソートの進行段階を示すパラメータで，SEQ＝1，2，……に対応して，ワークファイルがSWKA-XX，SWKB-XX，……と変わっていきます。コアソートの段階ではSEQ＝1です。

(2) ソート済みファイルのマージ

ソート済み小規模ファイルのなかから2つを選び，これを指定されたキーでマージすれば，2倍の長さの1つのファイルになります。マージ終了後は，元の2つのファイルをKILLしてしまいます。これをすべてのソート済みファイルについて行います。このような手順をくり返していけば，最後は元ファイルと同じ長さのソート済みファイルになります。これを具体的なフローチャートにしたのが**図3(b)**です。

マージの入力ファイルはSWK□-XXとSWK□-YYで，□にはSEQ=1,2,3,……に対応してA, B, C，……を入れます。XX, YYは最初それぞれ1，2を，次に3,4を，…と対応させます。マージ出力ファイルは，SWK△-ZZで，△にはSEQ+1に対応する

図3　シーケンシャルファイルのソートのフローチャート



(a) コアソート部分　　(b) マージ部分

英字を，ZZ は最初は 1，次に 2，……と対応させます。この初期設定をフローの最初に行います。

ソート済みファイル数 NF は一連のファイルをマージして一段階を終わった所で，ほぼ半減していきます。NF＝1 になればソートは完了です。NF＝0 は元ファイルのデータ件数が 0 の場合の例外です。NF＝1 の場合，そのファイルは目的とするソート出力ファイルですから，ファイル名を SWK□-XX というワークファイル名から指定されたソート出力ファイル名に変更してソートを終了します。

NF が 1 より大きい場合，出力ファイルの番号 ZZ を 1 だけ進め，SWK□-XX と SWK□-YY をマージしたファイル SWK△-ZZ を作成します。マージ終了後は元の 2 つのファイルを KILL します。

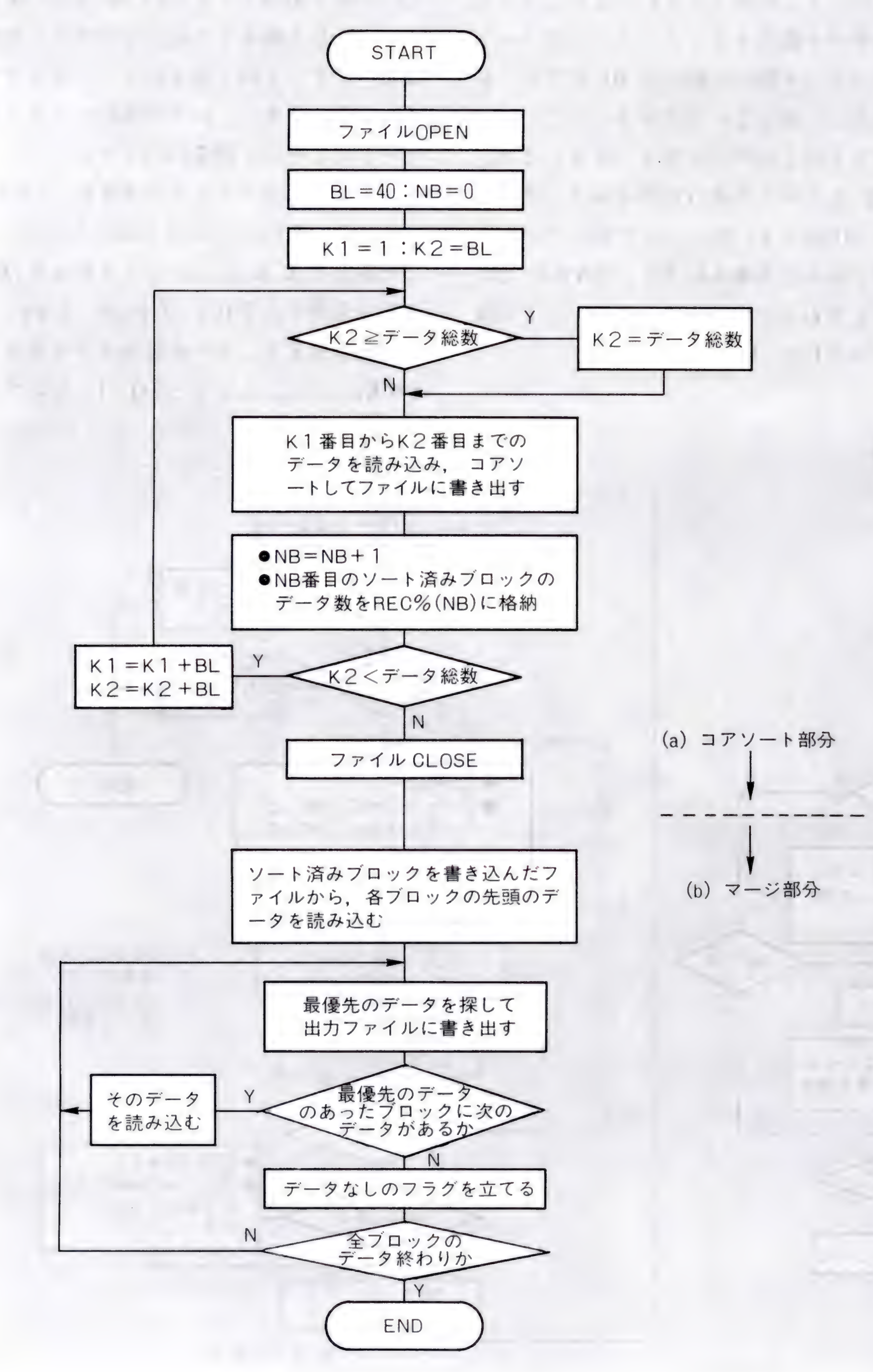
次に XX と YY を 2 ずつ進めて，また同様な処理をするわけですが，この場合にはファイルが 2 つ以上あるか，1 つしかないか，あるいはまったく残っていないかを判断してからにしなければなりません。

2 つ以上残っている場合（NF≧MFB）は，マージすべきファイルの番号 XX，YY を 2 ずつ，マージ出力ファイルの番号 ZZ を 1 だけ進めて再びマージ処理をします。

1 つしか残っていない場合（MFB＞NF≧MFA）は，この残っているファイルをマージ出力ファイルとして置き換えます。すなわち，ZZ を 1 だけ進め，ファイル名 SWK□-XX を SWK△-ZZ につけかえてしまいます。

まったく残っていない場合（NF＜MFA）は，この進行段階でのソート済みファイルのマージは完了ということですので，ファイル数 NF をマージ出力ファイルの数 ZZ に置き換え，次の段階へ進む意味で SEQ に 1 を加えて，再び一連のマージ部分の処理をくり返します。

図4　ランダムファイルのソートのフローチャート



5 ランダムファイルのソートの手順

ランダムファイルのソートの場合もシーケンシャルファイルのときと同様に，コアソートとマージの組み合わせです。異なる点は，ランダムファイルの場合，ソート済みのデータレコード群を必ずしも別のファイルにする必要はない（入力ファイルを保存する必要がなければ，元のフロッピディスクエリアに書き込んでしまえばよい）ことと，マージするときにこの各々のデータレコード群をマージ入力ファイルとみなして，

一度にマージしてしまうことの2つです。

ランダムファイルのソート手順のフローチャートを図4に示します。それでは流れにそって説明しましょう。

(1)ソート済みレコード群の作成

シーケンシャルファイルの場合と同様に，メモリに読み込んで並べかえるデータの件数をBLとし，その値を40にとります。

まず元ファイルをOPENし，データレコードをBL個だけメモリに読み込み，指定されたソートキーの順序に並べかえます。ランダムファイルの場合，BL個のデータの読み込みはK1，K2の2つのパラメータで行います。すなわち，最初にK1＝1，K2＝BLにセットし，以後はK1，K2にそれぞれBLを加えていくものとし，K1番目からK2番目までのデータレコードを読み込みます。K2がファイルに含まれるデータ総数より大きくなった場合は，K2はデータ総数に等しくとります。

このようにして読み込まれ並べかえられたBL個（一般にはK2−K1+1個）のデータは，元ファイルを保存するなら別のファイルの，保存しないなら同じファイルのやはりK1番目からK2番目のデータとして書き込みます。NBはソート済みデータのブロックの数でシーケンシャルファイルのソートの場合のNFに相当します。REC％(NB)はNB番目のソート済みでブロックに含まれるデータレコードの数です。REC％(NB)は，一般にはBL（＝40）に等しく，最後のブロックについてのみBLより小さい場合があります。

このようにして，元のファイルのデータレコードをBLずつ区切り，そのなかではソートされている形のファイルができあがります。

(2)ソート済みレコード群のマージ

(1)の処理によって作成されたファイルの各ブロックの先頭のデータ，すなわち，1，BL＋1，2＊BL＋1，……番目のデータレコードを読み込みます。1つのファイルからこのように飛び飛びのデータを読み込めるのがランダムファイルの特徴です。これらのデータのなかから指定されたソートキーの最優先のものを探し，出力ファイルに書き出します。

この最優先のデータのあったソート済みブロックからは次のデータレコードを取り出し，再び各ブロックから取り出したデータについて指定されたソートキーの最優先のものを探し，処理をくり返します。

あるブロックについて取り出すべきデータがなくなってしまったら，データなしのフラグを立て，このブロックは比較の対象からはずします。すべてのブロックについてこのフラグが立ったならば，ソート済みレコード群のマージ処理は終わり，ランダムファイルのソートは完了します。

6 ソートプログラムジェネレータ“SORTGEN”

ソートプログラムのジェネレータを**リスト2**に示します。ソート処理の手順についてはこれまでに詳しく説明しましたので，

リスト2●ソートプログラムジェネレータ“SORTGEN”

```
10 '    'SORTGEN'      ﾌｧｲﾙ Sort ﾌﾟﾛｸﾞﾗﾑ ｼﾞｪﾈﾚｰﾀ
20 CLEAR 5000:CLS:WIDTH 80,25
30 DIM HENSU$(20),QHENSU$(20),SA(20),SB(20),SC$(20),SD$(20)
40 COLOR 4:PRINT TAB(5),' ﾌｧｲﾙ Sort ﾌﾟﾛｸﾞﾗﾑ ｼﾞｪﾈﾚｰﾀ ':COLOR 7:PRINT
50 '
60 '    Step 1)  ｲｯﾊﾟﾝ ﾃﾞｰﾀ ｲﾝﾌﾟｯﾄ
70 '
80 INPUT 'ﾌﾟﾛｸﾞﾗﾑ ﾌｧｲﾙ ｷﾊﾞﾝ = ',O$
90 INPUT 'ﾌﾟﾛｸﾞﾗﾑ ﾒｲ          = ',PROMEI$
100 LINEINPUT 'ﾌﾟﾛｸﾞﾗﾑ ﾉ ｾﾂﾒｲ     = ',C$
110 PRINT:LINEINPUT 'ｲﾝﾌﾟｯﾄ ﾌｧｲﾙ ﾒｲ  =';FI$
120 LINEINPUT 'ｱｳﾄﾌﾟｯﾄ ﾌｧｲﾙ ﾒｲ =';FO$:PRINT
130 INPUT 'ﾌｧｲﾙ ｺｳｾｲ (R/SR)  = ',SR$:IF SR$='R' OR SR$='SR' THEN 140 ELSE 130
140 IF SR$='SR' THEN INPUT 'ﾃﾞｰﾀ ﾚｺｰﾄﾞ ﾉ ﾅｶﾞｻ = ',SRLEN
150 '
160 '     Step 2)  SORT ﾃﾞｰﾀ ｲﾝﾌﾟｯﾄ
170 '
180 PRINT:INPUT 'Sort ｷｲ ｺｳﾓｸ ｽｳ  =';NKII:PRINT
190 FOR I=1 TO NKII
200 PRINT 'Sort ｷｲ ('+STR$(I)+') ﾍﾝｽｳﾒｲ, ｶｲｼｲﾁ, ｼﾞｭﾝｼﾞｮ = ';:INPUT HENSU$(I),SA(
I),SC$(I)
210 IF SC$(I)='<' THEN SD$(I)='>' ELSE SD$(I)='<'
220 IF RIGHT$(HENSU$(I),1)='$' THEN INPUT '               ｹﾀｽｳ =',SB(I)
230 QHENSU$(I)='Q'+HENSU$(I):NEXT I
240 PRINT:INPUT 'ｲﾝﾌﾟｯﾄ ﾌｧｲﾙ ﾎｿﾞﾝ ｽﾙｶ (Y/N) ';YN$:IF YN$<>'Y' AND YN$<>'N' THEN
240
250 INPUT 'ﾌﾛｯﾋﾟｲ ｿｳﾁ ﾉ ﾀﾞｲｽｳ   (1/2) ';DAI%:IF DAI%<>1 AND DAI%<>2 THEN 250
260 '
270 '    Step 3)  ﾌﾟﾛｸﾞﾗﾑ ｽﾃｰﾄﾒﾝﾄ ｻｸｾｲ
280 '
```

```
290 OPEN "O",#1,O$+":"+PROMEI$
300 PRINT #1,"10 '     "+CHR$(34)+PROMEI$+CHR$(34)+"    "+C$
310 PRINT #1,"15 CLEAR 15000:CLS:WIDTH 80,25:COLOR 4:LOCATE 20,0:PRINT "+CHR$(34
)+"ﾌｧｲﾙ "+FI$+" ﾉ ｿｰﾄ"+CHR$(34)+":COLOR 7:PRINT"
320 IF SR$="SR" THEN 340 ELSE 1030
330 '
340 '     Step 4S)  ｼｰｹﾝｼｬﾙ ﾌｧｲﾙ ｿｰﾄ ｽﾃｰﾄﾒﾝﾄ ｻｸｾｲ
350 '
360 PRINT #1,"20 DIM SWA$(50)"
370 PRINT #1,"30 I%=0:SEQ=1:BL=40:SRLEN="+STR$(SRLEN)
380 PRINT #1,"40 PRINT "+CHR$(34)+"    ｺｱｿｰﾄ ﾌﾞﾛｯｷﾝｸﾞ ="+CHR$(34)+"+STR$(BL)"
390 PRINT #1,"50 T1=TIME:PRINT "+CHR$(34)+"SORT START AT "+CHR$(34)+"+TIME$"
400 PRINT #1,"60 OPEN "+CHR$(34)+"I"+CHR$(34)+",#1,"+CHR$(34)+FI$+CHR$(34)
410 PRINT #1,"70 IF EOF(1) THEN 100"
420 PRINT #1,"80 IF I%<BL THEN 90 ELSE 100"
430 PRINT #1,"90 I%=I%+1:A$=INPUT$(SRLEN+2,1):SWA$(I%)=LEFT$(A$,SRLEN):GOTO 70"
440 PRINT #1,"100 '    Core-sorting"
450 PRINT #1,"110 NS=I%:IF NS>1 THEN GOSUB 600:GOTO 130"
460 PRINT #1,"120 IF NS>0 THEN 130 ELSE 170"
470 PRINT #1,"130 NF=NF+1:GOSUB 400"
480 PRINT #1,"140 OPEN "+CHR$(34)+"O"+CHR$(34)+",#2,FO$"
490 PRINT #1,"150 FOR I=1 TO NS:PRINT #2,SWA$(I):NEXT I"
500 PRINT #1,"160 CLOSE #2:PRINT "+CHR$(34)+"NF = "+CHR$(34)+"+STR$(NF)"
510 PRINT #1,"170 IF EOF(1) THEN CLOSE #1 ELSE I%=0:GOTO 90"
520 PRINT #1,"180 T2=TIME:PRINT "+CHR$(34)+"Core Sort End at "+CHR$(34)+"+TIME$+
"+CHR$(34)+",  ("+CHR$(34)+"+STR$(NF)+"+CHR$(34)+" files )"+CHR$(34)+""
530 IF YN$="N" THEN PRINT #1,"190 KILL "+CHR$(34)+FI$+CHR$(34)
540 IF YN$="Y" AND DAI%=2 THEN PRINT #1,"190 BEEP:COLOR 6:PRINT:PRINT "+CHR$(34)
+"0 ﾊﾞﾝ ﾉ ﾌﾛｯﾋﾟｲ ｦ ｲﾚｶｴ, Return key ｦ ｵｼﾃｸﾀﾞｻｲ "+CHR$(34)+":COLOR 7":PRINT #1,"1
95 IF INKEY$<>CHR$(13) THEN 195"
550 PRINT #1,"200 '    Merge"
560 PRINT #1,"210 MFA=1:MFB=2:ZZ=0"
570 PRINT #1,"220 IF NF=0 THEN PRINT "+CHR$(34)+"ﾃﾞｰﾀ ｹﾝｽｳ = 0 !"+CHR$(34)+":GOT
O 270"
580 PRINT #1,"230 IF NF>1 THEN 280"
590 PRINT #1,"240 '    Sort End"
600 IF DAI%=1 THEN PRINT #1,"250 FMA$="+CHR$(34)+"SWK"+CHR$(34)+"+CHR$(64+SEQ)+"
+CHR$(34)+"-1"+CHR$(34)+""
610 IF DAI%=2 THEN PRINT #1,"250 Q1$=MID$(STR$(SEQ MOD 2),2,1):FMA$=Q1$+"+CHR$(3
4)+":SWK"+CHR$(34)+"+CHR$(64+SEQ)+"+CHR$(34)+"-1"+CHR$(34)
620 PRINT #1,"260 NAME FMA$ AS "+CHR$(34)+FO$+CHR$(34)
630 PRINT #1,"270 T3=TIME:PRINT "+CHR$(34)+"SORT END AT "+CHR$(34)+"+TIME$"
640 PRINT #1,"271 PRINT "+CHR$(34)+"  Merge Time ="+CHR$(34)+"+STR$(T3-T2)+"+CHR
$(34)+"sec,  SORT TOTAL ="+CHR$(34)+"+STR$(T3-T1)+"+CHR$(34)+"sec"+CHR$(34)+""
650 PRINT #1,"272 END"
660 PRINT #1,"280 '    Merge Routine"
670 PRINT #1,"290 ZZ=ZZ+1:GOSUB 500"
680 PRINT #1,"300 PRINT "+CHR$(34)+"SEQ ="+CHR$(34)+"+STR$(SEQ)+"+CHR$(34)+"
  ZZ ="+CHR$(34)+"+STR$(ZZ):GOSUB 700"
690 PRINT #1,"310 KILL FMA$:KILL FMB$:MFA=MFA+2:MFB=MFB+2"
700 PRINT #1,"320 IF NF>=MFB THEN 290"
710 PRINT #1,"330 IF NF>=MFA THEN ZZ=ZZ+1:GOSUB 500:NAME FMA$ AS FMC$"
720 PRINT #1,"340 NF=ZZ:SEQ=SEQ+1:GOTO 210"
730 PRINT #1,"400 '    File Name Sub  (1)"
740 IF DAI%=2 THEN Q$="1:" ELSE Q$=""
750 PRINT #1,"410 FO$="+CHR$(34)+Q$+"SWK"+CHR$(34)+"+CHR$(64+SEQ)+"+CHR$(34)+"-"
+CHR$(34)+"+MID$(STR$(NF),2)"
760 PRINT #1,"420 RETURN"
770 PRINT #1,"500 '    File Name Sub  (2)"
780 IF DAI%=1 THEN PRINT #1,"505 Q1$=CHR$(48):Q2$=Q1$"
790 IF DAI%=2 THEN PRINT #1,"505 Q1$=MID$(STR$(SEQ MOD 2),2,1):Q2$=MID$(STR$((SE
Q+1) MOD 2),2,1)"
800 PRINT #1,"510 FMA$=Q1$+"+CHR$(34)+":SWK"+CHR$(34)+"+CHR$(64+SEQ)+"+CHR$(34)+
"-"+CHR$(34)+"+MID$(STR$(MFA),2)"
810 PRINT #1,"520 FMB$=Q1$+"+CHR$(34)+":SWK"+CHR$(34)+"+CHR$(64+SEQ)+"+CHR$(34)+
"-"+CHR$(34)+"+MID$(STR$(MFB),2)"
820 PRINT #1,"530 FMC$=Q2$+"+CHR$(34)+":SWK"+CHR$(34)+"+CHR$(65+SEQ)+"+CHR$(34)+
"-"+CHR$(34)+"+MID$(STR$(ZZ),2)"
830 PRINT #1,"540 RETURN"
840 PRINT #1,"600 '    Core Sort Subroutine"
```

```
850 PRINT #1,"610 FOR I=1 TO NS"
860 PRINT #1,"620 FOR J=NS-1 TO I STEP -1"
870 LN=630:DR1$="SWA$(J)":DR2$="SWA$(J+1)":GOSUB 1730
880 PRINT #1,"670 GOSUB 1000:IF O%=2 THEN SWAP SWA$(J),SWA$(J+1)"
890 PRINT #1,"680 NEXT J:NEXT I"
900 PRINT #1,"690 RETURN"
910 PRINT #1,"700 '   Merge Subroutine"
920 PRINT #1,"710 OPEN "+CHR$(34)+"I"+CHR$(34)+",#1,FMA$:OPEN "+CHR$(34)+"I"+CHR
$(34)+",#2,FMB$:OPEN "+CHR$(34)+"O"+CHR$(34)+",#3,FMC$"
930 PRINT #1,"720 A$=INPUT$(SRLEN+2,1):B$=INPUT$(SRLEN+2,2)"
940 LN=730:DR1$="A$":DR2$="B$":GOSUB 1730
950 PRINT #1,"750 GOSUB 1000:IF O%=1 THEN PRINT #3,LEFT$(A$,SRLEN):GOTO 760 ELSE
 PRINT #3,LEFT$(B$,SRLEN):GOTO 770"
960 PRINT #1,"760 IF EOF(1) THEN 780 ELSE A$=INPUT$(SRLEN+2,1):GOTO 730"
970 PRINT #1,"770 IF EOF(2) THEN 790 ELSE B$=INPUT$(SRLEN+2,2):GOTO 730"
980 PRINT #1,"780 PRINT #3,LEFT$(B$,SRLEN):IF EOF(2) THEN 800 ELSE B$=INPUT$(SRL
EN+2,2):GOTO 780"
990 PRINT #1,"790 PRINT #3,LEFT$(A$,SRLEN):IF EOF(1) THEN 800 ELSE A$=INPUT$(SRL
EN+2,1):GOTO 790"
1000 PRINT #1,"800 CLOSE:RETURN"
1010 GOTO 1630
1020 '
1030 '    Step 4R) ﾗﾝﾀﾞﾑ ﾌｧｲﾙ ｿｰﾄ ｽﾃｰﾄﾒﾝﾄ ｻｸｾｲ
1040 '
1050 PRINT #1,"20 DIM B$(80),C$(80),IX%(80),MFL%(80),REC%(80)"
1060 PRINT #1,"30 '   Input File Open"
1070 PRINT #1,"40 OPEN "+CHR$(34)+"R"+CHR$(34)+",#1,"+CHR$(34)+FI$+CHR$(34)+":FI
ELD #1,255AS K$,1AS L$"
1080 PRINT #1,"50 KENSU=LOF(1):BL=40:NB=0"
1090 PRINT #1,"60 PRINT "+CHR$(34)+"ﾃﾞｰﾀ ｹﾝｽｳ ="+CHR$(34)+"+STR$(KENSU)+"+CHR$(3
4)+"   ｺｱｿｰﾄ ﾌﾞﾛｯｷﾝｸﾞ ="+CHR$(34)+"+STR$(BL)"
1100 PRINT #1,"70 T1=TIME:PRINT "+CHR$(34)+"SORT START AT "+CHR$(34)+"+TIME$"
1110 PRINT #1,"100 '    BL ｹﾝﾂﾞﾂ Core Sort & Write-out"
1120 IF DAI%=1 AND YN$="Y" THEN PRINT #1,"110 OPEN "+CHR$(34)+"R"+CHR$(34)+",#2,
"+CHR$(34)+"SWKA-1"+CHR$(34)+":FIELD #2,255 AS KK$,1 AS LL$"
1130 IF DAI%=2 THEN PRINT #1,"110 OPEN "+CHR$(34)+"R"+CHR$(34)+",#2,"+CHR$(34)+"
1:SWKA-1"+CHR$(34)+":FIELD #2,255 AS KK$,1 AS LL$"
1140 PRINT #1,"120 K1=1:K2=BL"
1150 PRINT #1,"130 IF KENSU<=K2 THEN K2=KENSU"
1160 PRINT #1,"140 GOSUB 470:NB=NB+1:REC%(NB)=K2-K1+1"
1170 PRINT #1,"150 IF K2<KENSU THEN K1=K1+BL:K2=K2+BL:GOTO 130"
1180 PRINT #1,"160 CLOSE:T2=TIME:PRINT "+CHR$(34)+"Core Sort End at "+CHR$(34)+"
+TIME$+"+CHR$(34)+",  ("+CHR$(34)+"+STR$(NB)+"+CHR$(34)+" blocks )"+CHR$(34)+""
1190 IF DAI%=2 AND YN$="N" THEN PRINT #1,"170 KILL "+CHR$(34)+FI$+CHR$(34)
1200 IF DAI%=2 AND YN$="Y" THEN PRINT #1,"170 BEEP:COLOR 6:PRINT:PRINT "+CHR$(34
)+"0 ﾊﾞﾝ ﾉ ﾌﾛｯﾋﾟｲ ｦ ｲﾚｶI, Return key ｦ ｵｼﾃｸﾀﾞｻｲ "+CHR$(34)+":COLOR 7":PRINT #1,"
180 IF INKEY$<>CHR$(13) THEN 180"
1210 IF DAI%=1 AND YN$="Y" THEN FI$="0:SWKA-1"
1220 IF DAI%=2 THEN FI$="1:SWKA-1"
1230 PRINT #1,"190 OPEN "+CHR$(34)+"R"+CHR$(34)+",#1,"+CHR$(34)+FI$+CHR$(34)+":F
IELD #1,255 AS K$,1 AS L$"
1240 PRINT #1,"200 '    Merge-out File Open"
1250 PRINT #1,"210 OPEN "+CHR$(34)+"R"+CHR$(34)+",#2,"+CHR$(34)+FO$+CHR$(34)+":F
IELD #2,255AS X$,1AS Y$:L=0"
1260 PRINT #1,"220 '    Merge First Data Read-in"
1270 PRINT #1,"230 FOR I=1 TO NB:IX%(I)=1:GET #1,BL*(I-1)+1:B$(I)=K$:C$(I)=L$:NE
XT I"
1280 PRINT #1,"240 '    Minimum Data Search"
1290 PRINT #1,"250 MB=0:FOR I=1 TO NB"
1300 PRINT #1,"260 IF MFL%(I)>0 THEN 330"
1310 PRINT #1,"270 IF MB=0 AND MFL%(I)=0 THEN SAIX=I:MB=1:GOTO 330"
1320 LN=290:DR1$="B$(I)":DR2$="B$(SAIX)":GOSUB 1730
1330 PRINT #1,"320 GOSUB 1000:IF O%=1 THEN SAIX=I"
1340 PRINT #1,"330 NEXT I"
1350 PRINT #1,"340 '    Minimum Data Output"
1360 PRINT #1,"350 L=L+1:LSET X$=B$(SAIX):LSET Y$=C$(SAIX):PUT #2,L"
1370 PRINT #1,"360 '    Next Data Get"
1380 PRINT #1,"370 IX%(SAIX)=IX%(SAIX)+1"
1390 PRINT #1,"380 IF IX%(SAIX)>REC%(SAIX) THEN MFL%(SAIX)=1:GOTO 390 ELSE GET #
```

```
1,BL*(SAIX-1)+IX%(SAIX):B$(SAIX)=K$:C$(SAIX)=L$:GOTO 240"
1400 PRINT #1,"390 '   Data End Check"
1410 PRINT #1,"400 FOR I=1 TO NB"
1420 PRINT #1,"410 IF MFL%(I)=0 THEN 240"
1430 PRINT #1,"420 NEXT I"
1440 PRINT #1,"430 '   Sort End"
1450 PRINT #1,"440 CLOSE:T3=TIME:PRINT "+CHR$(34)+"SORT END AT "+CHR$(34)+"+TIME
$"
1460 PRINT #1,"445 KILL "+CHR$(34)+FI$+CHR$(34)
1470 PRINT #1,"450 PRINT "+CHR$(34)+"  Merge Time ="+CHR$(34)+"+STR$(T3-T2)+"+CH
R$(34)+"sec,  SORT TOTAL ="+CHR$(34)+"+STR$(T3-T1)+"+CHR$(34)+"sec"+CHR$(34)+""
1480 PRINT #1,"460 END"
1490 PRINT #1,"470 '   Core-Sort Subroutine"
1500 PRINT #1,"480 FOR K=1 TO K2-K1+1"
1510 PRINT #1,"490 GET #1,K1+K-1:B$(K)=K$:C$(K)=L$"
1520 PRINT #1,"500 NEXT K"
1530 PRINT #1,"510 FOR N=K2-K1 TO 1 STEP -1"
1540 PRINT #1,"520 FOR K=1 TO N"
1550 LN=530:DR1$="B$(K)":DR2$="B$(K+1)":GOSUB 1730
1560 PRINT #1,"550 GOSUB 1000:IF O%=2 THEN SWAP B$(K),B$(K+1):SWAP C$(K),C$(K+1)
"
1570 PRINT #1,"560 NEXT K"
1580 PRINT #1,"570 NEXT N"
1590 IF DAI%=1 AND YN$="N" THEN PRINT #1,"580 FOR K=1 TO K2-K1+1:LSET K$=B$(K):L
SET L$=C$(K):PUT #1,K1+K-1:NEXT K":GOTO 1610
1600 PRINT #1,"580 FOR K=1 TO K2-K1+1:LSET KK$=B$(K):LSET LL$=C$(K):PUT #2,K1+K-
1:NEXT K"
1610 PRINT #1,"590 RETURN"
1620 '
1630 '     Step 5) ｷｲ ﾋｶｸ ｻﾌﾞﾙｰﾁﾝ ｽﾃｰﾄﾒﾝﾄ ｻｸｾｲ
1640 '
1650 PRINT #1,"1000 '     ｷｲ ﾋｶｸ ｻﾌﾞﾙｰﾁﾝ"
1660 FOR T=1 TO NKII
1670 PRINT #1,MID$(STR$(1010+2*T-1),2)+" IF "+HENSU$(T)+SC$(T)+QHENSU$(T)+" THEN
 O%=1:RETURN"
1680 PRINT #1,MID$(STR$(1010+2*T),2)+" IF "+HENSU$(T)+SD$(T)+QHENSU$(T)+" THEN O
%=2:RETURN"
1690 NEXT T
1700 PRINT #1,"1080 O%=1:RETURN"
1710 CLOSE:END
1720 '
1730 '   Sub A)  R,SR ﾌｧｲﾙ ｷｲ ｾｯﾄ Sub
1740 '
1750 FOR I=1 TO NKII:L=4:QA$="CVS(":QB$=")"
1760 IF RIGHT$(HENSU$(I),1)="%" THEN L=2:QA$="CVI("
1770 IF RIGHT$(HENSU$(I),1)="#" THEN L=8:QA$="CVD("
1780 IF RIGHT$(HENSU$(I),1)="$" THEN L=SB(I):QA$="":QB$=""
1790 PRINT #1,MID$(STR$(LN+I-1),2)+" "+HENSU$(I)+"="+QA$+"MID$("+DR1$+","+STR$(S
A(I))+","+STR$(L)+")"+QB$+":"QHENSU$(I)+"="+QA$+"MID$("+DR2$+","+STR$(SA(I))+","
+STR$(L)+")"+QB$
1800 NEXT I
1810 RETURN
```

リスト2のBASICチェックサム

LINE	+0	+1	+2	+3	+4	+5	+6	+7	SUM
10	00	FD	25	97	00	00	00	EA	:A3
90	71	F9	59	D8	A2	54	00	00	:91
170	00	DA	6C	32	52	41	4E	F5	:4E
250	8E	00	00	00	20	15	9A	ED	:4A
330	00	00	00	4E	7D	D9	D1	9E	:13
410	9F	4E	1E	01	D0	82	F0	E7	:35
490	F8	F2	69	A7	CD	57	36	DB	:2F
570	79	7D	29	CD	7E	7E	50	30	:68
650	9F	44	38	2B	7E	5F	75	14	:AC
730	75	18	B6	A3	77	4B	FB	A1	:44
	23	E9	88	32	A1	84	9F	11	/9B

LINE	+0	+1	+2	+3	+4	+5	+6	+7	SUM
810	A4	87	A6	EB	06	58	6A	9D	:21
890	56	AC	8B	5C	CC	86	F7	5E	:90
970	63	80	7E	55	03	00	00	00	:B9
1050	6C	79	4E	58	54	D3	52	89	:8D
1130	C1	2C	9B	C9	5F	BC	00	49	:B5
1210	41	0F	C4	0D	92	9A	71	36	:F4
1290	2B	85	19	A8	18	6B	72	86	:EC
1370	C6	F8	E7	FF	F2	81	6B	2A	:AC
1450	FF	BC	2F	9E	FD	EE	6C	6C	:4B
1530	69	B5	19	6D	72	76	FB	10	:97
	24	55	A4	7C	93	57	68	2F	/1A

LINE	+0	+1	+2	+3	+4	+5	+6	+7	SUM
1610	AB	00	00	00	80	77	7C	20	:3E
1690	93	F2	87	00	00	00	FB	A3	:AA
1770	A2	C3	C3	88	E0	00	00	00	:90
0	00	00	00	00	00	00	00	00	:00
0	00	00	00	00	00	00	00	00	:00
0	00	00	00	00	00	00	00	00	:00
0	00	00	00	00	00	00	00	00	:00
0	00	00	00	00	00	00	00	00	:00
0	00	00	00	00	00	00	00	00	:00
0	00	00	00	00	00	00	00	00	:00
	E0	B5	4A	88	60	77	77	C3	/78

リスト2の“SORTGEN”についてはこれを踏まえて簡単に紹介します。

(1)一般データインプット

これまでのジェネレータでも必ず最初に現れる部分です。作ろうとするソートプログラムに関する指定，ソート入力ファイルと出力ファイルの名称，RかSRかのファイル構成の区別，SRファイルの場合のデータレコードの長さの指定などを行います。

(2)ソートデータインプット

ソートの条件と環境を指定する部分です。ソートキー項目の数をまず指定し，各々の項目について変数名，データレコード中での開始桁位置，昇順か降順かの順序区別，さらに文字変数ならば桁数（長さ）を指定します。環境指定としては，インプットファイルの保存の要否，フロッピディスクの台数を与えます。

(3)プログラムステートメント作成

プログラム名と説明文のステートメントと，画面にソート実行を表示するステートメントを作ります。その後はシーケンシャルファイルとランダムファイルとで処理が異なる部分が多いので，ファイル構成に応じて（4S）あるいは（4R）に分岐します。

(4)ファイル構成別ステートメント作成

(4S) シーケンシャルファイルソートステートメント作成

以下で（ ）の中の数字は**リスト2**の行番号を表します。初期処理として，コアソート用の配列SWA$（50）をとり，各種パラメータの初期値をセットし，画面に必要な表示をしてソート入力ファイルをOPENするステートメントを作成します（360—400）。

OPENしたファイルからBL件ずつデータをメモリ内部に読み込み(410—430)，これをコアソートする（440—460）ステートメントを作ります。実際のコアソートのサブルーチンは840—900行で作られます。コアソート済みの配列を小規模ファイルとして書き出し(470—500)，インプットファイルが終わりか否かに応じた処理(510—520)をするステートメントを作成します。コアソート完了時点で，指定されたソート環境（入力ファイル保存か否かとフロッピディスクの台数）に応じたステートメントを作ります（530—540）。

次いでマージ部分の作成に入ります。まずマージ処理に必要な初期設定をし，マージ終了（したがってソート処理終了）を判断するステートメントを作ります（560—580）。マージ終了の場合は，ファイル名を指定された出力ファイル名につけかえ，メッセージを表示して終わります（590—650）。マージ処理のステートメントは660—720行で作成しますが，具体的なマージ処理のサブルーチンの作成は910—1000行で作られます。

コアソートされた配列をファイルへ書き出すときに必要なファイル名作成のサブルーチンは730—760行で，マージ処理の際に必要な3つのファイルのファイル名作成のサブルーチンは770—830行で作られます。

以上のメインステートメントを作り終わったら，R，SRファイルに共通なキー比較サブルーチンの作成（Step 5）へ分岐します。

(4R)ランダムファイルソートステートメント作成

初期処理として，必要な配列を定義し，ソート入力ファイルをOPENします。各種パラメータの値をセットし，画面に必要な表示をするステートメントを作成します(1050—1100)。

OPENしたファイルからBL件ずつデータを読み込み，これをコアソートしてファイルに書き出すステートメントを作成します(1110—1170)。実際の具体的な処理のステートメントは1490—1610行でサブルーチンとして作られます。入力ファイルデータのコアソート処理完了の時点で，指定されたソート環境（入力ファイル保存か否かとフロッピディスクの台数）に応じたステートメントを作成します（1180—1230）。

次いでマージ部分の作成に入ります。まずマージ入力ファイルをOPENし，ソート済みの各ブロックから最初のデータを読み込むステートメントを作成します（1240—1270）。これらのデータのなかから，ソートキー項目の最優先のものを探し（1280—1340），これを出力ファイルに書き出し(1350—1360)，そのブロックから次のデータを読み込む（1370—1390）ステートメントをそれぞれ作り出します。次に，各ブロックのデータがすべて空になったかどうかをチェックするステートメントを作り(1400—1430)，最後にソート処理終了の表示などのステートメントを作成します。

(5)キー比較サブルーチンステートメント作成

指定されたソート条件にしたがって，2つのデータレコードのキー項目を比較し，優先順位を決定するステートメントを作成します。これは前回のマージプログラムジェネレータの章で詳しく説明しましたので，そちらを参照してください。

Sub（A） S，SRファイルキーセット

データレコードからソートキー項目の値を取り出してセットするステートメントを作るサブルーチンです。データレコード1の変数名としては入力された変数名を用い，データレコード2の変数名はこの頭にQをつけたものを用いることは前回説明したとおりです。

7 テストデータの作成

“SORTGEN”の使用法の説明に入る前にソート用テストデータを作成します。現在ソートしたいデータファイルをお持ちの

方はそれを使って“SORTGEN”を使用していただければよいのですが，そうでない方はこれから述べるテストデータか，あるいはそれを変形したテストデータを作って“SORTGEN”を走らせてください。

テストデータはレコード長さ30のSRファイルとし，ファイル名をSAMPLE-Fとします。ファイルフォーマットを図5，ファイル作成プログラムを**リスト3**，ファイルのリストプログラムを**リスト4**に示します。1—4桁に単精度数値変数，11—12桁に整数，21桁に文字変数がいずれも乱数発生関数により作成され記入されており，他は空白となっているファイルで，データ件数を200件にとります。

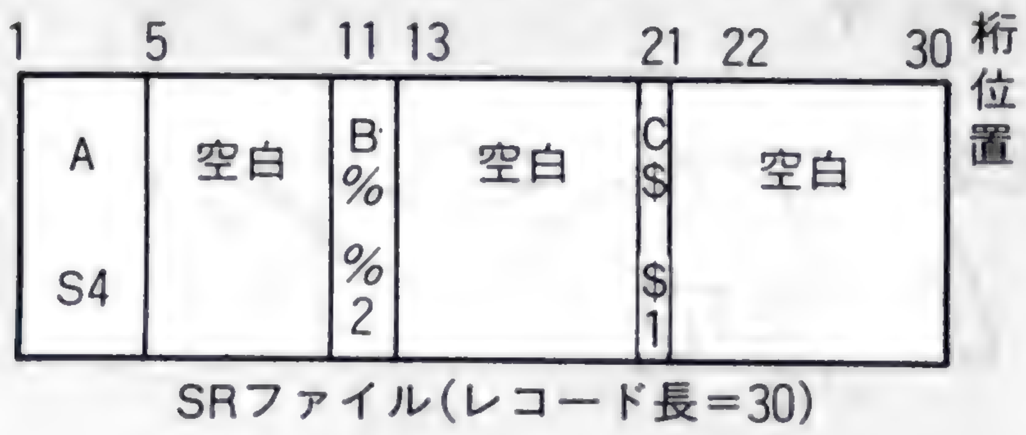
それでは**リスト3**のプログラムを実行させ，**リスト4**のプログラムで実際にファイルが作られたことを確認してください。このテストデータファイルをソートした後では，ファイル名はSAMPLE-Fから別のものに変えられますが，図4のプログラムの入力ファイル名をそれに置き換えればファイルフォーマットはまったく同じですので，ソート後のファイルをリストすることができます。

8 “SORTGEN”の使用法

それでは実際にソートプログラムを作成しましょう。まず**リスト2**の“SORTGEN”を打ち込みセーブしてください。FM-7, 8, 11のいずれでも結構ですが，フロッピディスクが少なくとも1台は必要です。カセットテープだけの場合は，BLの値を可能な限り大きくとりコアソートのみで終わるようにすれば，その範囲に限ってソートできます。空いているメモリが25.6Kバイトあるとして，1レコードの長さが256バイトで約100件，1レコードの長さが30バイトなら800件以上は可能ですから，カセットテープだけのシステムでもある程度実用になると思います。この場合は作成されたソートプログラムについてBLの値を修正してください。

ここでは前節で作成したテストデータファイルをソートするものとし，入力ファイルは保存，フロッピディスクは1台とした

図5　テスト用データファイルのフォーマット



SRファイル(レコード長＝30)

ソートキー　1.　A　昇順
　　　　　　2.　B%　降順
　　　　　　3.　C$　昇順

リスト3●テスト用ファイル作成プログラム

```
10 '  'SR-MAKE'
20 OPEN 'O',#1,'SAMPLE-F'
30 Q$=SPACE$(30)
40 FOR I=1 TO 200
50 C$=CHR$(64+26*RND(I)):MID$(Q$,21,4)=C$
60 B%=INT(5-10*RND(I)):MID$(Q$,11,2)=MKI$(B%)
70 A=INT(10-20*RND(I)):MID$(Q$,1,4)=MKS$(A):PRINT #1,Q$
80 NEXT I
90 CLOSE
```

リスト3のBASICチェックサム

```
LINE +0 +1 +2 +3 +4 +5 +6 +7 SUM
  10 00 62 F6 D3 2C 2F A7 88 :B5
  90 76 00 00 00 00 00 00 00 :76
---------------------------------
     76 62 F6 D3 2C 2F A7 88 /2B
```

リスト4●テスト用ファイルリストプログラム

```
10 '  'SR-DUMP'
20 OPEN 'I',#1,'SAMPLE-F'
30 IF EOF(1) THEN CLOSE:END ELSE Q$=INPUT$(32,1)
40 A=CVS(MID$(Q$,1,4))
50 B%=CVI(MID$(Q$,11,2))
60 C$=MID$(Q$,21,2):I=I+1:PRINT I,A,B%,C$:GOTO 30
```

リスト4のBASICチェックサム

```
LINE +0 +1 +2 +3 +4 +5 +6 +7 SUM
  10 00 5C 43 3C 87 33 00 00 :95
---------------------------------
     00 5C 43 3C 87 33 00 00 /95
```

図6　“SORTGEN”の入力画面の例

```
                    ﾌｧｲﾙ Sort ﾌﾟﾛｸﾞﾗﾑ ｼﾞｪﾈﾚｰﾀ

ﾌﾟﾛｸﾞﾗﾑ ﾌｧｲﾙ ｷﾊﾞﾝ = 0
ﾌﾟﾛｸﾞﾗﾑ ﾒｲ        = SORT-S
ﾌﾟﾛｸﾞﾗﾑ ﾉ ｾﾂﾒｲ    = SAMPLE-F Sort (ｲﾝﾌﾟｯﾄ ﾌｧｲﾙ ﾎｿﾞﾝ, ﾌﾛｯﾋﾟｲ 1ﾀﾞｲ)

ｲﾝﾌﾟｯﾄ ﾌｧｲﾙ ﾒｲ  =SAMPLE-F
ｱｳﾄﾌﾟｯﾄ ﾌｧｲﾙ ﾒｲ =OUTF

ﾌｧｲﾙ ｺｳｾｲ (R/SR)  = SR
ﾃﾞｰﾀ ﾚｺｰﾄﾞ ﾉ ﾅｶﾞｻ = 30

Sort ｷｲ ｺｳﾓｸ ｽｳ  =? 3

Sort ｷｲ ( 1) ﾍﾝｽｳﾒｲ, ｶｲｼｲﾁ, ｼﾞｭﾝｼﾞｮ = ? A,1,<
Sort ｷｲ ( 2) ﾍﾝｽｳﾒｲ, ｶｲｼｲﾁ, ｼﾞｭﾝｼﾞｮ = ? B%,11,>
Sort ｷｲ ( 3) ﾍﾝｽｳﾒｲ, ｶｲｼｲﾁ, ｼﾞｭﾝｼﾞｮ = ? C$,21,<
             ｹﾀｽｳ =1

ｲﾝﾌﾟｯﾄ ﾌｧｲﾙ ﾎｿﾞﾝ ｽﾙｶ (Y/N) ? Y
ﾌﾛｯﾋﾟｲ ｿｳﾁ ﾉ ﾀﾞｲｽｳ   (1/2) ? 1
```

例を用いて説明します。この例による入力画面を**図6**に示しますので，これを参照しながら以下の手順にしたがって操作してください。まず“SORTGEN”をRUNします。

1．プログラムを書き出す媒体の機番(0：，1：，CAS0：など）を入力します。

2．作成されるプログラムの名前を入力します。

3．作成されるプログラムの説明文をカタカナなどで入力します。

4．ソート入力ファイル，ソート出力ファイルの各々について，ファイル名を装置の機番を含めて入力します。

5．ファイル構成がRかSRのいずれであるかを入力します。SRファイルの場合は続いてデータレコードの長さをたずねてきますので，その値を入力します。

6．ソートキー項目の数を入力します。単純に1つの項目についてソートするときは1，**図5**の例では3です。

7．優先順位の高いキー項目から順に，6で指定したキー項目の数だけ次の入力を行います。

・キー項目変数名
・その項目のデータレコード内開始桁
・昇順降順指定
・文字変数の場合その長さ

変数名はタイプに応じて末尾に$,%,#などをつけます。昇順降順指定は，昇順なら<，降順なら>を入力します。

8．入力ファイルを保存するならY，しないならNを入力します。

9．フロッピディスクの台数1か2を入力します。カセットテープだけの場合は1を入力してください。

10．プログラムの作成が始まり，しばらくして画面に“Ready”が表示され完了します。

以上でソートプログラムジェネレータ“SORTGEN”の処理は終わり，指定した媒体にプログラムが作り出されます。**図6**の入力画面によって作成されたソートプログラムを**図7**に示します。これはシーケンシャルファイルのソートプログラムに相当しますので，**図3**のフローチャートと対比して理解を深めてください。

図6と同様な入力画面でランダムファイルを仮定して作成されたソートプログラムを**図8**に示します。これは**図4**のフローチャートを対比して理解してください。

図7　図6の入力で作られたソートプログラム

```
10 '    'SORT-S'   SAMPLE-F Sort (ｲﾝﾌﾟｯﾄ ﾌｧｲﾙ ﾎｿﾞﾝ, ﾌﾛｯﾋﾟｲ 1ﾀﾞｲ)
15 CLEAR 15000:CLS:WIDTH 80,25:COLOR 4:LOCATE 20,0:PRINT 'ﾌｧｲﾙ SAMPLE-F ﾉ ｿｰﾄ':C
OLOR 7:PRINT
20 DIM SWA$(50)
30 I%=0:SEQ=1:BL=40:SRLEN= 30
40 PRINT '    ｺｱｿｰﾄ ﾌﾞﾛｯｷﾝｸﾞ ='+STR$(BL)
50 T1=TIME:PRINT 'SORT START AT '+TIME$
60 OPEN 'I',#1,'SAMPLE-F'
70 IF EOF(1) THEN 100
80 IF I%<BL THEN 90 ELSE 100
90 I%=I%+1:A$=INPUT$(SRLEN+2,1):SWA$(I%)=LEFT$(A$,SRLEN):GOTO 70
100 '    Core-sorting
110 NS=I%:IF NS>1 THEN GOSUB 600:GOTO 130
120 IF NS>0 THEN 130 ELSE 170
130 NF=NF+1:GOSUB 400
140 OPEN 'O',#2,FO$
150 FOR I=1 TO NS:PRINT #2,SWA$(I):NEXT I
160 CLOSE #2:PRINT 'NF = '+STR$(NF)
170 IF EOF(1) THEN CLOSE #1 ELSE I%=0:GOTO 90
180 T2=TIME:PRINT 'Core Sort End at '+TIME$+',  ('+STR$(NF)+' files )'
200 '    Merge
210 MFA=1:MFB=2:ZZ=0
220 IF NF=0 THEN PRINT 'ﾃﾞｰﾀ ｹﾝｽｳ = 0 !':GOTO 270
230 IF NF>1 THEN 280
240 '    Sort End
250 FMA$='SWK'+CHR$(64+SEQ)+'-1'
260 NAME FMA$ AS 'OUTF'
270 T3=TIME:PRINT 'SORT END AT '+TIME$
271 PRINT '  Merge Time ='+STR$(T3-T2)+'sec,  SORT TOTAL ='+STR$(T3-T1)+'sec'
272 END
280 '    Merge Routine
290 ZZ=ZZ+1:GOSUB 500
300 PRINT 'SEQ ='+STR$(SEQ)+'       ZZ ='+STR$(ZZ):GOSUB 700
310 KILL FMA$:KILL FMB$:MFA=MFA+2:MFB=MFB+2
320 IF NF>=MFB THEN 290
330 IF NF>=MFA THEN ZZ=ZZ+1:GOSUB 500:NAME FMA$ AS FMC$
340 NF=ZZ:SEQ=SEQ+1:GOTO 210
400 '    File Name Sub  (1)
410 FO$='SWK'+CHR$(64+SEQ)+'-'+MID$(STR$(NF),2)
420 RETURN
500 '    File Name Sub  (2)
505 Q1$=CHR$(48):Q2$=Q1$
510 FMA$=Q1$+':SWK'+CHR$(64+SEQ)+'-'+MID$(STR$(MFA),2)
520 FMB$=Q1$+':SWK'+CHR$(64+SEQ)+'-'+MID$(STR$(MFB),2)
530 FMC$=Q2$+':SWK'+CHR$(65+SEQ)+'-'+MID$(STR$(ZZ),2)
540 RETURN
600 '    Core Sort Subroutine
610 FOR I=1 TO NS
620 FOR J=NS-1 TO I STEP -1
630 A=CVS(MID$(SWA$(J), 1, 4)):QA=CVS(MID$(SWA$(J+1), 1, 4))
631 B%=CVI(MID$(SWA$(J), 11, 2)):QB%=CVI(MID$(SWA$(J+1), 11, 2))
632 C$=MID$(SWA$(J), 21, 1):QC$=MID$(SWA$(J+1), 21, 1)
670 GOSUB 1000:IF O%=2 THEN SWAP SWA$(J),SWA$(J+1)
680 NEXT J:NEXT I
690 RETURN
700 '    Merge Subroutine
710 OPEN 'I',#1,FMA$:OPEN 'I',#2,FMB$:OPEN 'O',#3,FMC$
720 A$=INPUT$(SRLEN+2,1):B$=INPUT$(SRLEN+2,2)
730 A=CVS(MID$(A$, 1, 4)):QA=CVS(MID$(B$, 1, 4))
731 B%=CVI(MID$(A$, 11, 2)):QB%=CVI(MID$(B$, 11, 2))
732 C$=MID$(A$, 21, 1):QC$=MID$(B$, 21, 1)
750 GOSUB 1000:IF O%=1 THEN PRINT #3,LEFT$(A$,SRLEN):GOTO 760 ELSE PRINT #3,LEFT
$(B$,SRLEN):GOTO 770
760 IF EOF(1) THEN 780 ELSE A$=INPUT$(SRLEN+2,1):GOTO 730
770 IF EOF(2) THEN 790 ELSE B$=INPUT$(SRLEN+2,2):GOTO 730
780 PRINT #3,LEFT$(B$,SRLEN):IF EOF(2) THEN 800 ELSE B$=INPUT$(SRLEN+2,2):GOTO 7
80
790 PRINT #3,LEFT$(A$,SRLEN):IF EOF(1) THEN 800 ELSE A$=INPUT$(SRLEN+2,1):GOTO 7
90
800 CLOSE:RETURN
1000 '      ｷｲ ﾋｶｸ ｻﾌﾞﾙｰﾁﾝ
1011 IF A<QA THEN O%=1:RETURN
1012 IF A>QA THEN O%=2:RETURN
1013 IF B%>QB% THEN O%=1:RETURN
1014 IF B%<QB% THEN O%=2:RETURN
1015 IF C$<QC$ THEN O%=1:RETURN
1016 IF C$>QC$ THEN O%=2:RETURN
1080 O%=1:RETURN
```

9 実際のソート処理の例

7節で作成したテストデータファイルを**図7**のプログラムでソートしたときのCRT画面表示を**図9**に示します。総データ件数が200で，コアソートの1ブロックを40としていますので，5つのファイルが最初に作られます(NF＝1～5)。ここでコアソート部分が完了し246秒（4分6秒）を要しています。

続いてマージの第1段階（SEQ＝1）に入り，5本のファイルの中1と2および3と4がマージされ（ZZ＝1，ZZ＝2)，5はそのままでファイル名だけつけかえられています。第2段階（SEQ＝2）ではマージ入力ファイルは3つですから，マージ処理は一度だけ行われ(ZZ＝1)，3つ目のファイルはやはりファイル名だけつけかえられます。第3段階（SEQ＝3）ではマージ入力ファイルは2つとなり，この処理を終わって(ZZ＝1)，ソートは完了です。このマージ部分の処理に132秒を要し（Merge Time)，全体でソートに要した時間は378秒です（SORT TOTAL)。

また，**図7**の20行の配列の大きさと30行のBLの値をそれぞれ300にして同様な処理を行ったところ，マージ処理部分はなく，ソート所要時間は1125秒でした。

カセットテープだけのシステムの場合はこのようにコアソートのブロッキングの大きさを変え，さらに次の変更をします。

60行 “SAMPLE-F” → “CAS0：SAMPLE-F”
130行 GOSUB400→ INPUT “テープヲトリカエテカラ，ニンイノキーヲオシテクダサイ”，S$
140行 FO$→ “CAS0：OUTF”
250，260行 DELETE

テストデータファイルのソートで，コアソートのブロッキングの大きさを40と300(実際は200）としたときの差は大きいものです。実際に処理する場合は，そのファイルの長さ，一般的なデータ件数など状況を考慮してソート所要時間を最小にするようにBLの値を変えてください。

図8　図6と同様の入力で作成されたソートプログラム(ランダムファイル用)

```
10 '    'SORT-R'   Random File Sort Sample (ｲﾝﾌﾟｯﾄ ﾎｿﾞﾝ, ﾌﾛｯﾋﾟｲ 1 ﾀﾞｲ)
15 CLEAR 15000:CLS:WIDTH 80,25:COLOR 4:LOCATE 20,0:PRINT 'ﾌｧｲﾙ SAMPLE-F ﾉ ｿｰﾄ':C
OLOR 7:PRINT
20 DIM B$(80),C$(80),IX%(80),MFL%(80),REC%(80)
30 '    Input File Open
40 OPEN 'R',#1,'SAMPLE-F':FIELD #1,255AS K$,1AS L$
50 KENSU=LOF(1):BL=40:NB=0
60 PRINT 'ﾃﾞｰﾀ ｹﾝｽｳ ='+STR$(KENSU)+'    ｺｱｿｰﾄ ﾌﾞﾛｯｷﾝｸﾞ ='+STR$(BL)
70 T1=TIME:PRINT 'SORT START AT '+TIME$
100 '    BL ｹﾝﾂﾞﾂ Core Sort & Write-out
110 OPEN 'R',#2,'SWKA-1':FIELD #2,255 AS KK$,1 AS LL$
120 K1=1:K2=BL
130 IF KENSU<=K2 THEN K2=KENSU
140 GOSUB 470:NB=NB+1:REC%(NB)=K2-K1+1
150 IF K2<KENSU THEN K1=K1+BL:K2=K2+BL:GOTO 130
160 CLOSE:T2=TIME:PRINT 'Core Sort End at '+TIME$+',  ('+STR$(NB)+' blocks )'
190 OPEN 'R',#1,'0:SWKA-1':FIELD #1,255 AS K$,1 AS L$
200 '    Merge-out File Open
210 OPEN 'R',#2,'OUTF':FIELD #2,255AS X$,1AS Y$:L=0
220 '    Merge First Data Read-in
230 FOR I=1 TO NB:IX%(I)=1:GET #1,BL*(I-1)+1:B$(I)=K$:C$(I)=L$:NEXT I
240 '    Minimum Data Search
250 MB=0:FOR I=1 TO NB
260 IF MFL%(I)>0 THEN 330
270 IF MB=0 AND MFL%(I)=0 THEN SAIX=I:MB=1:GOTO 330
290 A=CVS(MID$(B$(I), 1, 4)):QA=CVS(MID$(B$(SAIX), 1, 4))
291 B%=CVI(MID$(B$(I), 11, 2)):QB%=CVI(MID$(B$(SAIX), 11, 2))
292 C$=MID$(B$(I), 21, 1):QC$=MID$(B$(SAIX), 21, 1)
320 GOSUB 1000:IF O%=1 THEN SAIX=I
330 NEXT I
340 '    Minimum Data Output
350 L=L+1:LSET X$=B$(SAIX):LSET Y$=C$(SAIX):PUT #2,L
360 '    Next Data Get
370 IX%(SAIX)=IX%(SAIX)+1
380 IF IX%(SAIX)>REC%(SAIX) THEN MFL%(SAIX)=1:GOTO 390 ELSE GET #1,BL*(SAIX-1)+I
X%(SAIX):B$(SAIX)=K$:C$(SAIX)=L$:GOTO 240
390 '    Data End Check
400 FOR I=1 TO NB
410 IF MFL%(I)=0 THEN 240
420 NEXT I
430 '    Sort End
440 CLOSE:T3=TIME:PRINT 'SORT END AT '+TIME$
445 KILL '0:SWKA-1'
450 PRINT '  Merge Time ='+STR$(T3-T2)+'sec,  SORT TOTAL ='+STR$(T3-T1)+'sec'
460 END
470 '    Core-Sort Subroutine
480 FOR K=1 TO K2-K1+1
490 GET #1,K1+K-1:B$(K)=K$:C$(K)=L$
500 NEXT K
510 FOR N=K2-K1 TO 1 STEP -1
520 FOR K=1 TO N
530 A=CVS(MID$(B$(K), 1, 4)):QA=CVS(MID$(B$(K+1), 1, 4))
531 B%=CVI(MID$(B$(K), 11, 2)):QB%=CVI(MID$(B$(K+1), 11, 2))
532 C$=MID$(B$(K), 21, 1):QC$=MID$(B$(K+1), 21, 1)
550 GOSUB 1000:IF O%=2 THEN SWAP B$(K),B$(K+1):SWAP C$(K),C$(K+1)
560 NEXT K
570 NEXT N
580 FOR K=1 TO K2-K1+1:LSET KK$=B$(K):LSET LL$=C$(K):PUT #2,K1+K-1:NEXT K
590 RETURN
1000 '      ｷｲ ﾋｶｸ ｻﾌﾞﾙｰﾁﾝ
1011 IF A<QA THEN O%=1:RETURN
1012 IF A>QA THEN O%=2:RETURN
1013 IF B%>QB% THEN O%=1:RETURN
1014 IF B%<QB% THEN O%=2:RETURN
1015 IF C$<QC$ THEN O%=1:RETURN
1016 IF C$>QC$ THEN O%=2:RETURN
1080 O%=1:RETURN
```

図9　実際のソート処理のCRT画面

```
                    ﾌｧｲﾙ SAMPLE-F ﾉ ｿｰﾄ

  ｺｱｿｰﾄ ﾌﾞﾛｯｷﾝｸﾞ = 40
SORT START AT 00:44:59
NF =  1
NF =  2
NF =  3
NF =  4
NF =  5
Core Sort End at 00:49:05,  ( 5 files )
SEQ = 1       ZZ = 1
SEQ = 1       ZZ = 2
SEQ = 2       ZZ = 1
SEQ = 3       ZZ = 1
SORT END AT 00:51:17
  Merge Time = 132sec,  SORT TOTAL = 378sec
```

FMぽーと

Logo解説書2冊

2冊のLogoの解説書,『パソコンLOGOプログラミング』と『FM Logoガイドブック V1.0』があいついで発刊された。

『パソコンLOGOプログラミング』は,FM Logoを中心に,Logoを使いこなしたい人を対象とした本。Logoの基本であるリスト処理と再帰に重点をおき,数値計算と数式処理,再帰による複雑な図形の描画,リスト処理機能の応用などを,20～100行程度の手頃な例題プログラムで解説している。

著者の小谷善行は人工知能の研究者で,LOGOコンパイラを学会に発表したこともある。発行は東海大学出版会(〒160 東京都新宿区新宿3-27-4新宿東海ビル☎03-356-1541)。A5版,224ページ。定価2,400円。

『FM LogoガイドブックV1.0』は,初心者向けのタートルグラフィックに多くのページを割いた,FM Logo(V1.0)のマニュアルに基づいた入門書。重点事項を書き出してあったり,大きめの活字を使ったりして,わかりやすいように工夫してある。著者はテクニカルライタの須藤智香。発行は共立出版㈱(〒112 東京都文京区小日向4-6-19☎03-947-2511)。B5版,139ページ。定価1,800円。

3Mディスクがラインアップ

住友スリーエム㈱(〒158 東京都世田谷区玉川台2-33-1☎03-709-8526)では,信頼性,耐久性を向上させたフロッピディスクを,8インチ10種類,5インチ4種類を新発売した。標準小売価格は5インチ2Dが1,700円,2HDが2,300円。

ブラザーから低価格プリンタ M-1009

ブラザー販売㈱では,このたび49,800円という低価格の超小型インパクトドットマトリックスプリンタM-1009を発売した。

M-1009はFMシリーズに対応したプリンタで,このほか,PC・MSXに対応した(モード切換)M-1009Xも発売されている。

9ピンドットマトリックスプリンタのヘッドとしては世界最小の超小型ヘッドを採用,333(W)×190(D)×70(H)mm,約3.0kgと,小型・軽量を実現した。主な仕様は次のとおり。

●印字速度=約50cps ●改行時間=250ms ●紙送り方式=フリクションフィード,スプロケットホイール(オプション) ●用紙=カット紙,ロール紙,ファンフォールド紙(10インチ) ●紙送りピッチ=1/6, 1/8, n/72, n/216インチ

また,漢字印字などのユーティリティプログラム「プリンタ機能拡張BASIC」もオプションで発売された。KANJI "3A7B 213C"のように16進コードを文字列に入れるだけで簡単に漢字が印字できる。文字列中に「=」「*」などの制御文字を入れれば,倍角,縦書き,文字間隔,行送り,オフセット(印字開始桁指定),反復などの指定もでき,漢字プリンタに変身させてしまう。このほか,FM-7の裏RAMを利用したプリンタスプーラ機能,ハードコピーの領域指定(キャラクタ単位)などもBASICのコマンドとして拡張されている。FM-7用,テープベース(ディスクにセーブ可)。4,000円。

プレゼント情報

今月号のプレゼントを紹介します。ご希望の方はとじこみのアンケートハガキに必要事項を記入のうえ,お送りください。締め切りは9月10日(必着)。当選者は11月号(10月18日発売)誌上で発表します。

①超小型プリンタ　M-1009　1名
ブラザー提供

左で紹介した超小型インパクトドットマトリックスプリンタ。プリンタ機能拡張BASICとセットで。

②OS-9ユーティリティコマンドVol.1　5名
星光電子提供

SOFT REVIEWで紹介したもので,レベル1,2とも使用可能。

③スタコラCRUSH　(T, 5D)　各5名

SOFT REVIEWで紹介した,反射神経+パズル型のゲーム。T, 5Dを明記。

④パソコンLOGOプログラミング　5名

左で紹介したFM中心のLogoの解説書です。

⑤Oh!FM　Tシャツ(青・黄)　各10名

Oh!FMのロゴの入ったオリジナルTシャツ。サイズ(L, M)と色を明記。

⑥Oh!FM 9月号ノート　10名

毎度おなじみOh!FMノート。表紙は本誌と同じ体裁で,中はなにも印刷してありません。

7月号プレゼント当選者

①英単語学習　5名
前田祐介(名古屋市)・高橋生(厚岸郡)・篠原正人(新宿区)・小島保太郎(横須賀市)・長谷川弘幸(大阪市)
②重要文型学習　5名
森井眞紀子(名古屋市)・高野明(世田谷区)・大石憲治(呉市)・花輪健司(御殿場市)・渡辺泰良(北区)
③GAME SOFT SELECT 4　5名
門尾孝是(立川市)・竹内務(海部郡)・西野浩司(川崎市)・岩崎俊人(江東区)・松尾哲二(四日市市)
④データガード　10名
坪田幸夫(新治郡)・志田洋子(宗像市)・池防芳一(豊岡市)・竹中智巳(大垣市)・眞鍋和子(上川郡)・中田朗(八代市)・赤司智昭(大分市)・中村智一(勢多郡)・郷勝彦(三島郡)・日巡武彦(鎌倉市)
⑤漢字ROM　1名
横道正和(諏訪郡)

虫つくろいのページ

本誌8月号記事中,次の誤りがありました。読者ならびに関係各位に深くおわびするとともに訂正いたします。

●8月号　「F-BASICで拡張RAMの入出力を!」p.105 リスト4

```
140 WHILE ASC(KRINPUT$(65536!*CAUNTO+65529!,1))=0
170 PRINT (ASC(KRINPUT$(65536!*CAUNTO+65529!,1))-32)*&H1000;" bytes free"
```

P. 98　3-141 KRLOAD 拡張RAM ロード

P.102　形式 KRPRINT %領域番号,AT,アイテム番号,オフセット,式

補足・リスト1～4を打ち込む前に必ずCLEAR, &H1FFFをダイレクトに実行してください。

READERS' AREA

▶残暑お見舞申し上げます。世の中リバイバルブームで，ウルトラマンやゴジラや大魔人が再び脚光を浴びていますが，我がFM-8もいつの日にかリバイバルで脚光を浴びるのではないか，と確信しております。以上お中元でした。(高松市・池田純子・23)

――これはこれは大層なお中元。まことにありがとうございます。

▶ううぅ…うれしいです！ FM-11用のゲームありがとう。でも拡張RAMの記事は貧乏人にはとてもつらい記事です。だって先月やっとの思いで漢字ROMを買ったんだもの。奥様にオネダリして。(東久留米市・野中佳男・28)

――やはり，ゲームもやりたい11ユーザはたくさんいました。現在リーダーズサービスの申し込み率を7と11が争っています。

▶先日，ある大学の公開講座で『コンピュータ入門』というのがあったので行ってきました。感想―大型コンピュータのOSもすばらしいけど，わが愛機FMのOS-9はもっとすばらしい。(岡山市・佐藤博之・28)

――今月の特集いかがでした？

▶マイコン歴とは

1．興味を持ちはじめたときから
2．BASICを勉強しはじめたときから
3．BASICをマスタしたときから
4．マイコンを買ったときから

1～4のどれでしょうか？ (柏市・渡辺隆一・15)

――すべてです。たとえナイコン族でも，マイコン歴10年と書いてもおかしくありません。なん年と判断するのはご自身です。

▶ぼくは今，じん臓の病気で病院に入院しています。とうぜんFM-7とはわかれています。入院してから十日ほどですがFM-7とそれだけ長くはなれていると，やりたくてしようがない。でも，できないものはできないので，Oh！FMを一日に何十回もくり返して読んでいます。Oh！FMにはふしぎな力があるのか，何回読んでもあきません。(札幌市・青木敢斗・11)

――何度も読んでくれてありがとう。早く元気になってください。

▶要望。1，会話型デバッグツール(LIST "ABC"でABCを使用している行のみ表示，FIND "ABC"でカーソルがABCに移動，⏎キーで次のABC行へいくなど)。2，バグさがし(コンパイラのようにRUNする前にバグを捜す。A＝A＋1の前にA＝0がないとか，B(S)のとき配列宣言がないとかを見つけてくれるもの)。この2つを開発して載せてもらえませんか。(中野区・赤井裕一・23)

――ほ，欲しい。これがあれば少しは質問電話が減る。いや絶対減る。

▶須納瀬先生，電気磁気学IIの単位をどうかください。お願いします。(船橋市・永野聡雄・21)

――あの～。お中元の宛先，間違っていませんか？

▶パソコン（ハード，ソフト）の後悔しない買い方。①マイコンショウ，データショウの直前は買わない（新製品の可能性）。②新製品は発売直後には買わない（バグやバージョンアップの可能性）。③情報を集める（売れゆき状況，シリアスな批判，活字にならない内輪の話を雑誌，業界関係，ショップ店員などから）。④周辺機器，ソフトの出具合を見る。⑤試用してみる。また試用させてくれる店を捜す。――これだけやれば完璧だ。(尼崎市・高橋光彦・29)

――たしかに，損をしない買物のコツは事前の検討にあります。ただ，業界関係の情報はなかなか手に入りませんし，また，いつが買い時かという見極めも難しいものです。

▶3.5インチと5インチのディスクソフトの出方はどうなってますか。77やディスクユニットを購入するときに重大な問題になると思うのですが。(光市・徳山大樹・14)

――各社の広告をご覧のとおり，5インチ版のものは順次3.5インチ版に移されていますし，新製品は5インチ，3.5インチ同時発売になっているケースが多いようです。なお，77への5インチの接続方法は10月号に掲載予定です。

▶宣誓！女房さま。私は（FM-7を買ったかわりに）今後，一切，飲み屋には足を踏み入れません？……ノドモト　スギレバ　ナントヤラ……。(美祢市・白石正春・35)

――空手形の返済はいつなさるのですか。

▶新製品問題について。ソフトを完全にパソコンメーカが出している状況ならともかく，第三者のソフトハウスの出すものがほとんどという現状では，ハードの変更はすぐにソフトの枯渇につながります。上位コンパチもこの点では同じことで「買い増し」という形でもう1つパソコンを買わなければいけなくなります（FM-8が証明しています）。この点でR.A.のEditor氏の見解は甘い!!と思います。(茨木市・伊藤清隆・26)

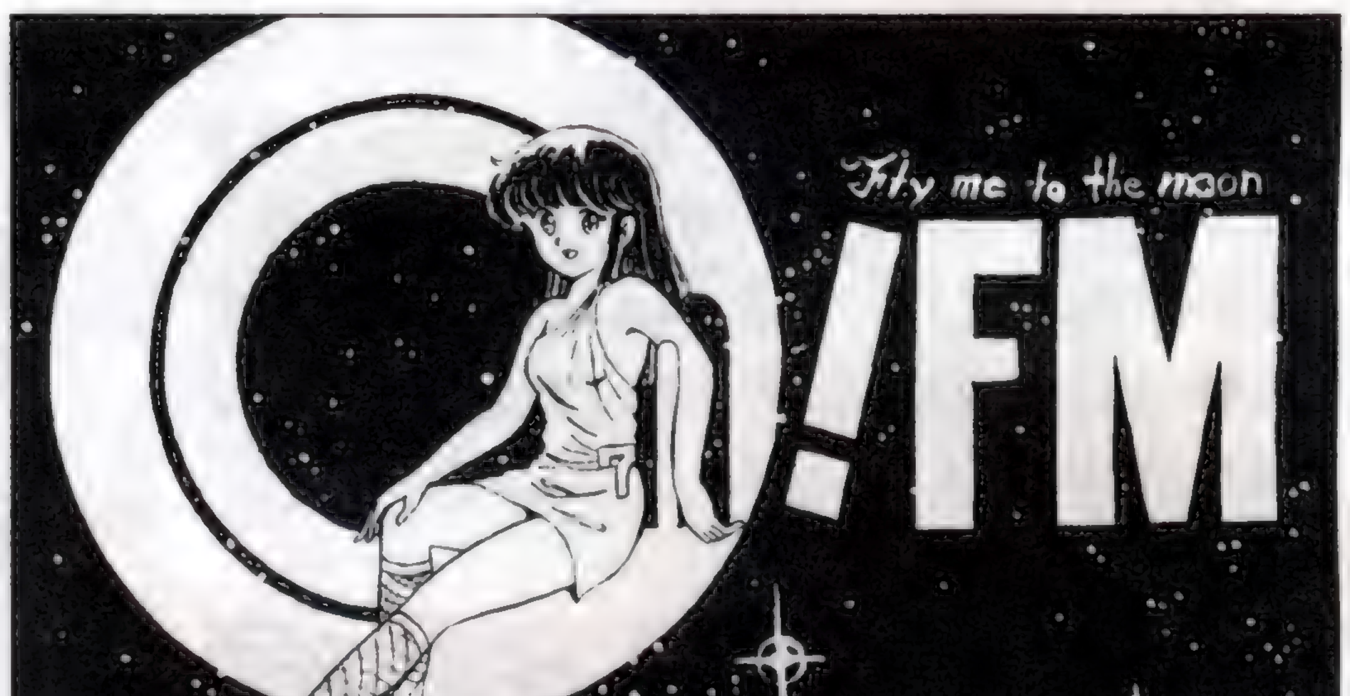
▶技術の進歩で，次から次へとレベルアップまたは新機種発表を耳にする最近ですが，そうした際にFMファンが，どのような自衛策で対処するか，いかに最初の精神を壊さずに確保できるかが問題だと思います。(京都市・近藤勉・34)

――苦労して手に入れたものは，いとおしいものです。パソコンでも同じで，自分のマシンを陳腐化させないためいろいろと努力するのは当然のことですが，マクロな視点も忘れてはいけないと思います。長期広範囲にわたる技術革新にどう対処していくかの方針がなければ変動に飲み込まれるだけです。自分はどうしていくのか充分に考えて，より望ましい使用環境が得られるよう，積極的に対応すべきだと僕は考えています。（何人かの方からまじめな意見をいただきましたが，2つのどちらかでした。新機種ユーザからの意見が届かず，公平に欠くことろがあるので今月の件へのコメントは控えました。）

▶新製品劇について一言。FM-77のユーザがFM-8,7のユーザの努力を感じてくれればすばらしいと思います。『77は一日にしてならず』。8のユーザさんありがとう（私は7のユーザです。）(東筑摩郡・百瀬喜代治・23)

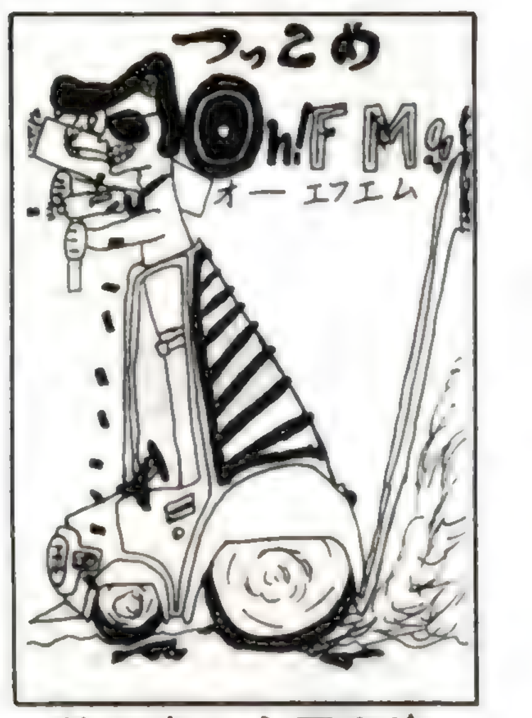
――新しいものは古いものの積み重ねの上に成り立っていることを感じ，その功績に感謝する。――忘れたくありませんね。

▶編集者の皆様のご苦労が分かります。毎回新機種が出るごとに，READERのおろかな知の病だれ，魚のグチではありませんか。しかし不満のハケロでもあるのです。頑張ってください。また，力を入れてください。だからこそいろいろと行間を読んでいます。(浜松



▲船橋市・中島隆

▲土浦市・富田剛



秩父市・永田和宏

入間郡・橋本利浩

青梅市・山中徹

東大阪市・上田裕之

市・長井淑郎・50）

——先月の編集後記に書きましたとおり，どんな原稿もお待ちしていますので，愚痴だろうがなんだろうがかまいません（良識の範囲内ですが，もちろん)。でも，ご覧のとおりのいたらぬ編者で，行間なんてものはありません（もっと勉強せにゃな〜)。

▶汎用コンピュータが256 Kだ512 Kだといっていたころからの付き合いです。当時と比べると，最近のパソコンの機能増大の様子はまさに隔世の感があります。今のパソコン少年たちが我々の年代になるころは，いったいどんな世の中になっているやら？（市原市・中村英亮・37）

——20年後というと，例のINSも完成しているころ。どれだけ社会が電子化されるのでしょうか。さて，どうなるのやら。

▶FM祭りのときのこと。アホな友人がおりまして，ドアドアの作者の中村光一さんにサインをねだっていました。中村さんはテレていました。（松戸市・菅原幸夫・15）

——イベントは一般向けの見世物であると同時に，業者間の交流の場でもあります。あのとき裏では名刺交換競争が激しく行われていました。ところで，中村さんには気がつきませんでした。おしいことをした。

▶IDカードの謎が，今明かされようとしている。——定期購読しよーっと！（奈良市・山科聡・21）

——あの申し込み案内のイラストも米村さんです。しかしSOFT BANKカードと四番町アドベンチャがどう関係あるのか僕は知りません。（まだ解けないんです。グスッ。）

▶もぐらのほりでエと一緒に登場する女の子の名前は？　もしないなら，ナナ子（7子）がいい。（8がはいるとハナ子，11がはいるとイヤナ子……。（熊谷市・吉田武史・19）

▶あの子の名前マイコにしません？（金沢市・勝田俊哉・18）

——もちろんあの絵は米村さんが描きました。①チーフのMに似ているとの話もありますが，作者，モデルとも否定しています。②名前は知りません。③マイコはM部長の娘の名です。

▶10 EXEC −512⏎　RUN⏎。なかなか使いがいのあるやつでしょう！（加古川市・満田博之・16）

——こういう過激なことはバックアップを取ってからしてください。悪用しないように！

▶アマグラマ講座はBASICの基本的な命令を使い，理解しながら，利用価値の高い，実用的なプログラムを作成するので，初心者には最適だ。終了後はセミグラマ講座を。（志木市・石川真・33）

▶現在OS-9で苦しんでいます。マニュアルは大変難しく，具体的イメージがつかめません。よく判っている人が書くとああなるんですね。（松江市・福桜盛一・44）

——「ゆりかごから墓場まで」これがOh！FMのモットーです。

▶R.A.を読むと，「同年代の所帯持ちがやっぱり苦労しているんだな」と胸が痛くなります。（文京区・高橋俊充・28）

——頑張ってください。

▶Readers' Angels である大和撫子の聡明な方々へのお願い——あなたの回りのマイコン狂をあたたかい目で見守ってください。男は大人になってもオモチャが好きなのです。ハイ！（柏市・鈴木昭・25）

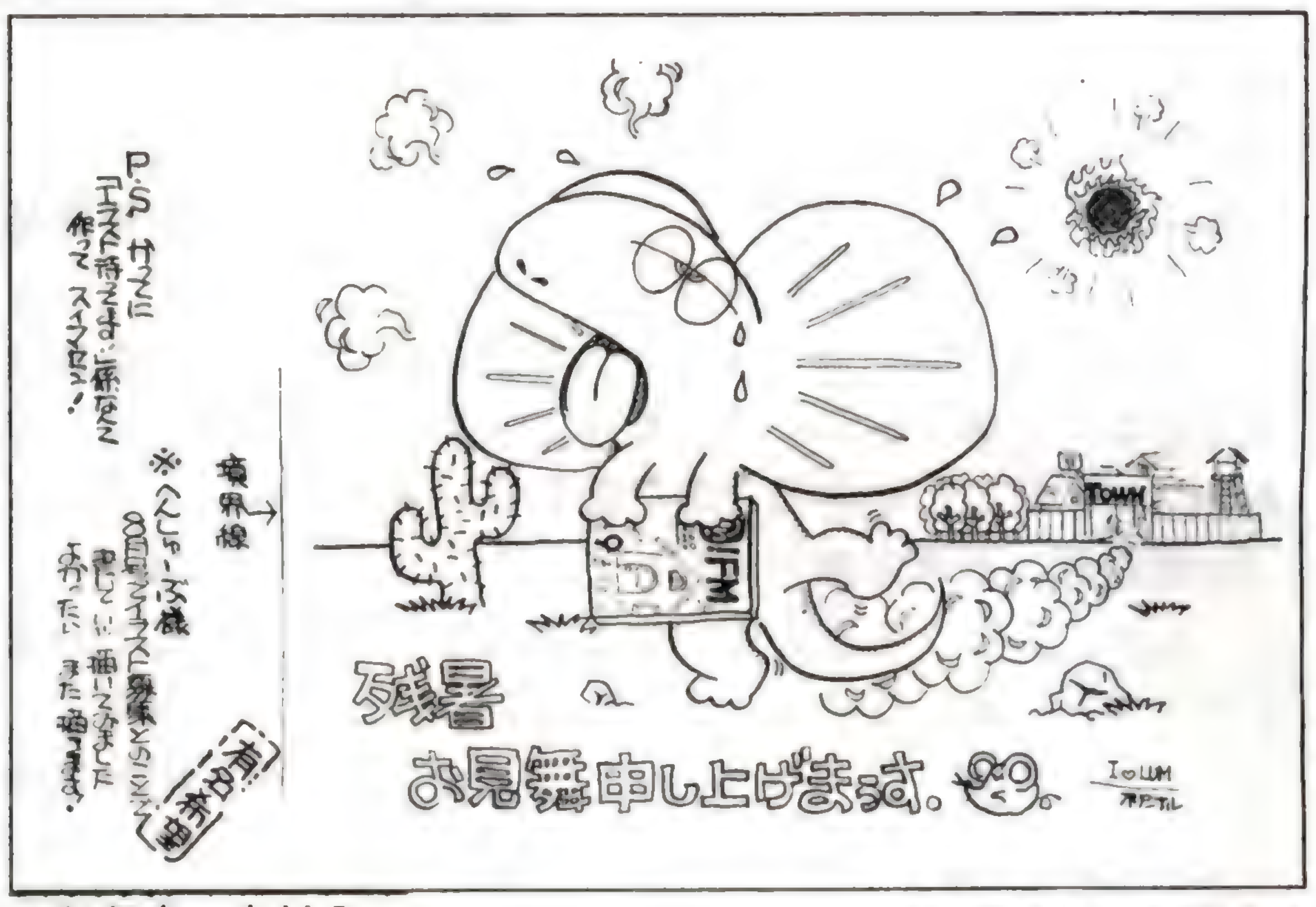
——いや，同じですよ。井戸端会議でサラシ物にされるのは。

▶最近話題の「M」「真理」について。「まつおか　まり(女)」なのか，「まさとし(男)」なのかいろいろ想像していましたが，一つ判ったので報告します。“老人”です。どうしてかは7月号編集後記の左最下段をお読みください。（札幌市・飯田弘昭・17）

——真理(しんり)は誰にもわかりません。

▶8月号の三田村さんへ，6人でパソコンクラブを作り，会報にプログラムのテクニックを発表したり，C.G.コンテストをしたり，初心者講座をしたり，けっこう楽しくやっています。人数がいると長いプログラムは打ち込みや作製の分担ができるので楽です。（所沢市・外塚桂・16）

▶高3のとき，友人と2人で創ったマイコンクラブ。一時は部員も20名を数え，文化祭に



▲加賀市・市村公一

▼横浜市・山川裕章

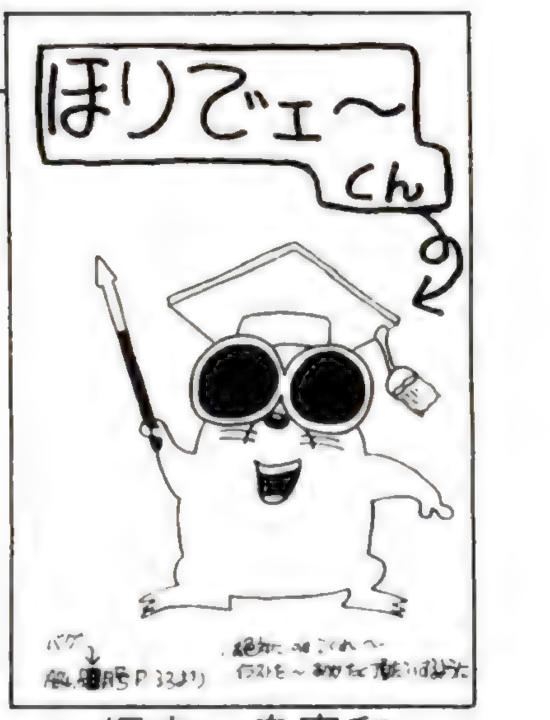
◀練馬区・永田孝博

◀四日市市・武藤毅

——先月「無競争で掲載」といった手前，締め切りに間に合ったイラストはすべて載せました。遅れて着いた人ゴメンナサイ。

加須市・鈴木覚

堺市・宮越和史



堺市・泉宏和

も参加したのですが……1年経ち，夏に卒業後初めて学校に行ってみると，活動を停止してしまった，我がクラブがあった。かなし〜。（豊中市・竹内公一・19）

——うまくいったり，いかなかったり，サークルの運営は難しいものです。苦労話をおよせください。来月はもっと行数をとります。

▶もうだめだ。しめきりにまにあわないよー。（世田谷区・瀬尾由美子・18）

——いや，まだ1時間ある。あとX枚なんとかするぞー。と，今日という日も暮れていく。

▲練馬区・小野田陽徳

▲京都市・丹野圭

EXPRESS

●仲間募集●

▶「ソフト・キャッツ」は全国的な富士通関係情報（特にFM-7,8）の情報交換サークルです。クラブの会長さん，ディスク所有の個人ユーザ，会社学校関係の人など，いろいろな人とコミュニケーションをはかりたいと考えています。返信用切手を同封してご連絡ください。会費は無料です。（〒143 東京都大田区大森中3-14-17 ☎03-763-9792 福島忠男）

▶「FMユーザーズサークル」では広く7，8ユーザを求めています。ビギナー，女性大観迎。希望者は住所氏名などを書き，60円切手を同封してお送りください。なお以前に入会して当方より連絡のない方，御一報ください。（〒899-26鹿児島県日置郡伊集院町麦生田1006☎09927-3-3011 福留耕一）

▶FM-11を持つ愛知県もしくは近県の方，FM-11のことをお話ししましょう。（〒450 愛知県名古屋市中村区稲葉地町8-91名幸荘2号 ☎052-411-8937 石田尚美）

▶AD2を買ったものの，初心者なものでうまく使えません。僕と情報交換してくれる仲間を求めます。（〒630 京都府京都市北区等持院南町60-10晴和寮☎075-463-4858 河口浩）

▶28歳のFM-7ユーザです。購入4ケ月，まだBASICもままなりませんが熱意はあります。誰かマイコンについてお話ししませんか。（〒270 千葉県松戸市三ケ月355-5 ☎0473-43-5688 白沢昌史）

▶OS-9ユーザでマニア以外の方，一緒に勉強しませんか。初心者も大歓迎です。現在OS-9で走るワープロを制作しています。（〒166 東京都杉並区高円寺南5-21-5-809シャンボール高円寺☎03-316-3284 北川誠二）

▶「PCUC」では会員募集をしています。現在会員数20名。往復ハガキでご連絡ください。（〒162 東京都新宿区矢来町156☎03-267-0279 雑賀英樹）

▶マイコンをやっている女の子，誰か文通しませんか。僕は高2です（〒596 大阪府岸和田市西大路町50-27 ☎0724-44-7007吉岡賢二）

▶宇都宮市内でFMシリーズをお持ちの方，自作ソフトや情報の交換をしませんか。（〒321-01栃木県宇都宮市宮ノ内4-237-5 ☎0286-53-5654 市原裕彦）

▶アドベンチャゲーム大好き少年（少女）。僕のところに連絡を。（〒851-04長崎県西彼杵郡三和町晴海台20-15 ☎0958-92-1129 柴田しんご）

▶いっしょにゲームソフトの開発をしませんか。（〒336 埼玉県浦和市文蔵4-21-17 ☎0488-63-8542 吉村潜）

▶誰かパソコン友達になってください。（〒933 富山県高岡市羽広193-29☎0766-21-8746 伊勢昌生）

▶FM-7ユーザの方，情報交換をしましょう。（〒069-15 北海道夕張郡栗山町富士8-29☎01237-2-4392 大谷信博）

▶FM-7ユーザの方，情報交換をしましょう。（〒072 北海道美唄市東2条南5丁目☎01266-2-5826 鈴木寿幸）

▶FM-7ユーザの方，情報交換をしましょう。（〒028-13岩手県下閉伊郡山田町飯岡10-1-15 上沢健一）

●案　内●

▶東海高校マイコン部では，10月におこなう記念祭にてロボットを展示します。名古屋付近の方，ぜひご来場ください。（〒458 愛知県名古屋市緑区有松町57-257☎052-621-7336 角田文彦）

▶城南高校化学部では去年に引き続き，FM-7による「基礎からの化学・第2版」を文化祭で展示する予定です。鳴門市付近の方，ぜひご来場ください。（〒772 徳島県鳴門市撫養町斎田☎08868-6-4889 泉泰正）

▶第2水準の漢字データを作成しましたので，希望者の方に，必要な漢字の部分だけリストをさしあげます。往復ハガキで連絡ください。（〒978 福島県石川郡石川町中野字鍛治内176-3 遠藤孝松）

投稿者一覧

▶6月15日から7月10日の間に届いた作品名を紹介します。Gはゲーム。Uはユーティリティ，実は実用ソフト，音は音楽ソフト，学は学習ソフトを示します。Pはプログラムの略です。

・G「ギャラクシー」中村健二18福生市・U「中間色見本作成ツール」園部信男　越谷市・U「DATA 文作成P」広瀬勲38世田谷区・U「シーケンシャルファイル管理P」松月章25伊勢市・実「ラジオ放送タイマー録音」小川陽一・上尾市・U「BASIC09 用ハードコピー」渡部剛士　東村山市・実「ピアノギターコードダイアグラム」千田雄一15佐世保市　・実「漢字住所録」高橋正則　京都市・U「BASIC チェックサムオンメモリ版」安田弘幸　大津市・G「THE LAST ONE」 内田聡信28春日部市・実「家計簿」重泉新一　札幌市・音「リズムパターン」 森雅人16札幌市・音「ハナカゲ」美野輪勝博61所沢市・U「高速SAVE」「横FILES」斉藤誠　千葉市・実「エアチェック用分秒計算P」安東友哉27大阪市・G「100 m ダッシュ」工藤康宏14名古屋市・U「マシン語入力スピードアップ」藤田慎一　堺市・学「タイプ練習」堀内啓士15名古屋市・G「数当てゲーム」中村秀生14奈良市・学「円周率の計算」林昭幸40高槻市・G「ハノイの塔」針原祥志郎32武生市・学「筆算計式の計算練習Ｖ2.0」玉城秀樹　糸満市・G「4目並べ」幡生安生　福井市・U「構造化リスト出力PV.1.0&V1.1」森茂　川越市・U「BASICチェックサムオンメモリ版」G「ブロック崩し」花高信哉15佐伯郡・U「PC-9801のCP/M-86の標準ディスクをFM-11で読む」大田正光　台東区・G「囲碁棋譜作成ＰＶ 2.0」大矢利夫39座間市・G「ロボットゲーム」木下修市14苫小牧市・G「2人用BINGO V2.0」中尾英己　新宿区・G「碁石取り」中出慶一39豊中市・G「F1グランプリ」 岡田仁・四日市市・U「OS-9FM-7·ROMRAM モード共用ＩＰＬ」本多明20米沢市・U「ラベルナンバ変換P」宮平圭一22三重郡・実「利子計算」学「NOAXI」玉城秀樹　糸満市・U「キャラクタジェネレータ」「マシン語モニタ」新井孝幸15横浜市・G「撃沈ゲーム」佐藤行威15入間郡・U「マシン語入力ツール」小島一敏17　千葉市・学習「分数トレーニング」石川真33志木市・U「マシン語モニタ」古川明　足立区・U「データサーチャ」成瀬健一18国分寺市・G「パズルゲーム」赤津雅明20練馬区・G「迷路」小塩誠次　葛飾区・G「SUBMARIN」吉田朋正　加古川市・学「CLOCK」石川善久30藤沢市・U「キャラクタで漢字を」江端上28松原市・U「DATA/REM MAKER」「Function plotter」市村鋭一22豊中市・G「上陸部隊を阻止せよ」 大西正芳　江戸川区・G「つらら」秋元克紀19江東区・音「FM-8 音楽演奏」「Rydeen」「テクノポリス」尾形航　横須賀市・音「君に胸キュン」実「バイオリズム」高坂武　横須賀市

ショートプログラム集

レーシングゲーム

岡田 仁, 高(三重県四日市市) for FM-7,8

高速スクロールゲームです。FMの単色モード時に青と赤の画面が消されずに，再び画面下方からそのまま現れることを利用したプログラムです。PRINT 文で改行するだけなので高速です。

RUN すると画面に青い車の列（敵）が現れ，緑色のタイトルと自分の車（赤色）が表示されます。その後キー入力待ちになるので，何かキーを押すとゲーム開始です。[1]で左に，[3]で右に移動するので，青い車をさけて，できるだけ先に進んでください。キーを離すと左右の移動は止まります。青い車と衝突するとスコアが表示され，ゲームオーバとなります。

〔編集部記〕

単色モード時の欠点を逆に利用してしまうという発想がよかったので採用しました。でも，この欠点がFM-11からはなくなってしまったので，11ユーザは遊べません。CONSOLE, , , 1はエラーにならないんですけどね。

FM-7用BASICチェックサム（□はFM-8用）

```
                         41        D8
LINE +0 +1 +2 +3 +4/ +5 +6 +7 \SUM
 100 00 AC 68 78 8B/ 3F 01 E9 :40
 180 64 AE B9 E8 [3E] CD 60 B7 :[D5]
 260 3C 75 8F 00 00 00 00 00 :40
---------------------------------
     A0 CF B0 60 [C9] 0C 61 A0 /[55]
                  CC             58
```

●レーシングゲーム●

©1984 H.&T. OKADA

FM-7用（FM-8は変更行を直してください）

```
100 ' ***** F1  GRAND PRIX   For FM-7,8     By  H. & T. Okada *****
110 WIDTH 80,25:CONSOLE 0,25,0,1:COLOR 7,0:CLS
120 CLEAR:DEFINT A-Z:DIM CAR(27),CSH(27):RANDOMIZE TIME/4:X=324
130 DEFFNT(X)=POINT(X+12,61)+POINT(X+12,78)
140 FOR I=0 TO 26:READ CAR(I),CSH(I):NEXT
150 FOR I=1 TO 181 STEP 60:FOR J=4 TO 580 STEP 64
160 R=INT(RND*2)*32+J:PUT@(R,I)-(R+23,I+17),CAR,PSET,1:NEXT J,I
170 SYMBOL(40,25),"F1  GRAND PRIX",5,3,4,,OR
180 PUT@(X,61)-(X+23,78),CAR,PSET,2:LOCATE 0,24:BEEP:K$=INPUT$(1)
190 PUT@(X,61)-(X+23,78),CAR,AND,1:K=VAL(INKEY$):S=S+1:PRINT
200 X=X-32*(K=1)*(X>32)+32*(K=3)*(X<608):IF FNT(X) THEN 220
210 PUT@(X,61)-(X+23,78),CAR,OR,2:FOR I=0 TO 8:BEEP1:BEEP0:NEXT:GOTO 190
220 FOR I=0 TO 30:PUT@(X,61)-(X+23,78),CSH,AND,1:FOR J=0 TO 100:NEXT
230 PUT@(X,61)-(X+23,78),CSH,OR,2:FOR J=0 TO 20:BEEP1:BEEP0:NEXT J,I
240 FOR J=0 TO 2000:NEXT:CONSOLE 0,25,0,0:CLS
250 SYMBOL(100,50),"****  G A M E  O V E R  ****",2,1,5
260 SYMBOL(180,70),"YOUR SCORE    "+STR$(S),2,1,5:K$=INKEY$:LOCATE 0,10:END
270 DATA 1023,387,-16381,-16360,-64,-30848,-1988,3150,8191,4174,-6145,2051,-61,
    1792,-7,-14591,-26209,-28276,768,-7461,-16369,-28189,15600,32515,3900,12779
280 DATA -4089,-14848,15584,-32400,921,-29193,-16383,-12488,-15488,-24826,231,
    -3085,120,-15392,32286,-15679,32743,1033,-392,6204,26142,6174,36,6460,15,
    17410,-16,6176
```

FM-8用変更行

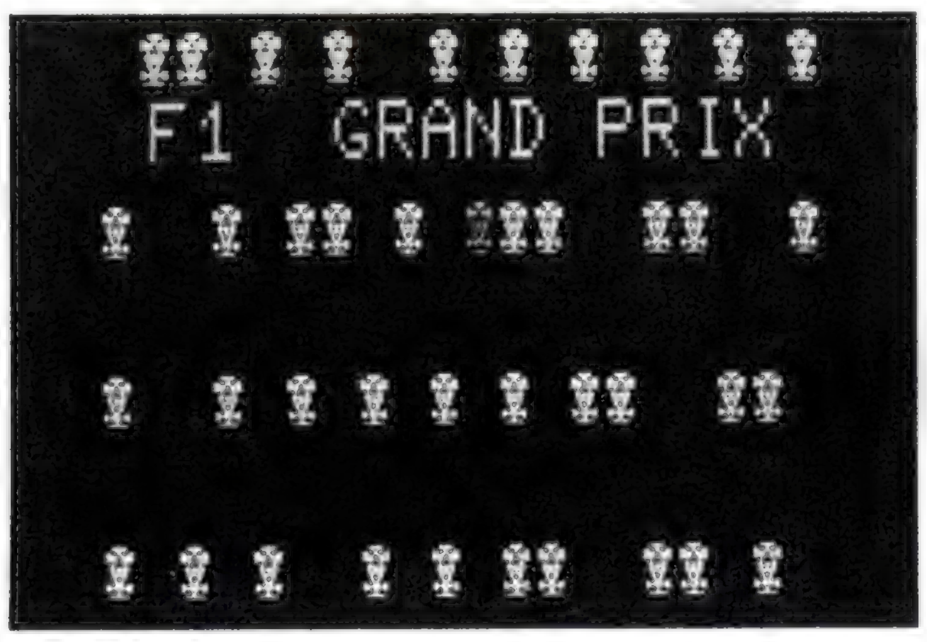
```
220 FOR I=0 TO 15:PUT@(X,61)-(X+23,78),CSH,AND,1:FOR J=0 TO 100:NEXT
```

サブマリンファイト

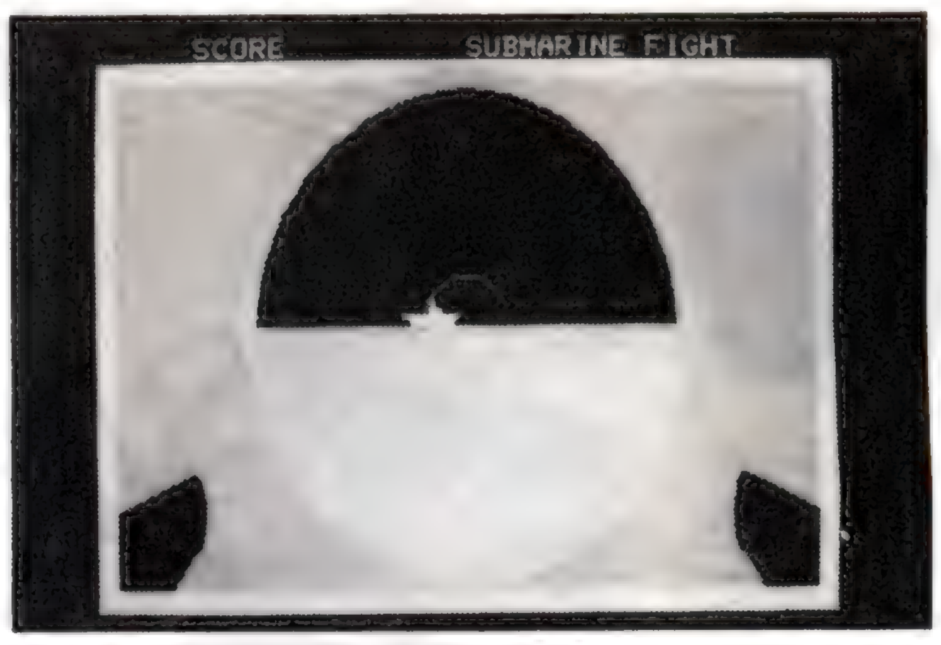
吉田 朋正(兵庫県加古川市) for FM-7

グラフィックに少し凝ったプログラムです。潜望鏡から見える敵の船をミサイルで撃破するゲームです。速い船をやっつけるほど高得点が得られます。ミサイルを多く撃つとゲームが早く終了してしまいます。

RUNすると画面が描かれ，そのままゲー



◀レーシングゲーム

サブマリンファイト▶


ムが始まります。タイミングよくスペースバーを押して撃破してください。使うキーはスペースバーのみです。タイムオーバにより終了します。

〔編集部記〕

ゲームそのものは，なかなかスピード感もあり音もダイナミックで，けっこう楽しめます。

ただ，リスト中にいくつか意味不明なところがあり，また，おせじにも読みやすいプログラムとはいえないようです。

とにかく動けばよい，という人もいるようですが，きれいなプログラムを書くように心がけたほうが，あとあと自分のためになると思います。

今後の研究課題というところでしょうか。

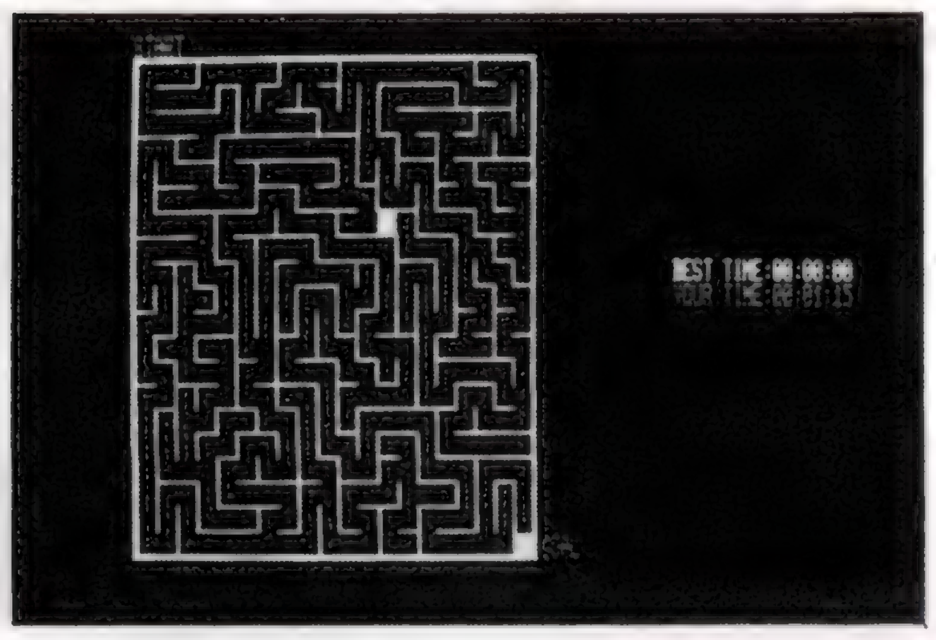
FM-7用BASICチェックサム

LINE	+0	+1	+2	+3	+4	+5	+6	+7	SUM
10	16	C3	F9	A6	40	71	C1	1E	:08
90	48	12	98	F5	C2	00	0E	99	:50
170	82	ED	0D	6A	31	67	14	60	:F2
250	DB	97	B8	8A	F0	60	61	5B	:C0
330	B8	0C	C2	00	00	00	00	00	:86
	73	65	18	8F	23	38	44	72	/90

●サブマリンファイト●

©1984 T.YOSHIDA

```
10 GOSUB150
20 SCREEN7,7:WIDTH40,25:PLAY"S8M1000O3DFL4CL2AEL8DL4FL2"
30 SCREEN7,7:FOR I=0 TO 7:COLOR=(I,I):NEXT:T=0:RANDOMIZETIME/4
40 S=0:C=2::SCREEN7,7:LOCATE5,0:COLOR5:PRINT"SCORE              SUBMARINE FIGHT"
50 LINE(0,1)-(39,24),"■",7,B:FOR I=100 TO 102:LINE(120,I)-(520,I),PSET,1:NEXT I
60 FOR I=102 TO 130 STEP 2:LINE(130,I)-(510,I),PSET,1:NEXT I
70 FOR I=130 TO 160 STEP 3:LINE(140,I)-(500,I),PSET,1:NEXTI:PAINT(320,20),5,1,7
80 CIRCLE(320,100),200,6,,,,N:CIRCLE(320,100),185,6,,,,N:PAINT(320,15),6,6
90 PAINT(320,20),0,6,1:PAINT(20,60),1,6,7:CIRCLE(320,100),40,4:LINE(320,130)-(32
0,110),PSET,4
100 LINE(260,100)-(290,100),PSET,4:LINE(350,100)-(380,100),PSET,4
110 LINE(16,165)-(80,152),PSET,0:LINE(60,191)-(85,175),PSET,0:CIRCLE(80,165),10,
0,1.5,,,F:PAINT(40,170),0,0,7
120 LINE(623,165)-(559,152),PSET,0:LINE(579,191)-(554,175),PSET,0:CIRCLE(559,165
),10,0,1.5,,,F:PAINT(599,170),0,0,7
130 LINE(320,70)-(320,90),PSET,4:PAINT(20,60),1,6,7:SCREEN2,7:GOTO 200
140 'ﾃｷ ﾉ ｳｺﾞｷ
150 COLOR7,0:CLS:LINE(0,5)-(50,8),PSET,7,BF:LINE(0,7)-(6,8),PSET,0,BF
160 CONNECT(46,8)-(50,6)-(50,8)-(46,8),0:LINE(24,3)-(34,4),PSET,7,BF
170 LINE(26,0)-(28,2),PSET,7,BF:LINE(22,0)-(26,0),PSET,7
180 DIM A% (116):GET@A(0,0)-(50,8),A%,G
190 T=T+1:IF T>299 THEN 320 ELSE RETURN
200 T=T+1:IF T>299 THEN 320 ELSE V=INT(RND(1)*4)+4:D=INT(RND(1)*7)+1:X=INT(RND(1
)*300)+120:A=INT(RND(1)*2+1):IF A=1 THEN 240 ELSE GOTO 220
210 COLOR=(C,D):PUT@(X,91)-(X+50,99),A%,PSET,C:IF INKEY$=" " THEN T=T+5:GOTO 260
 ELSE RETURN
220 X=X+V:IF X>454 THEN 200 ELSE GOSUB 210
230 PUT@(X,91)-(X+50,99),A%,PRESET:GOTO 220
240 X=X-V:IF X<150 THEN 200 ELSE GOSUB 210
250 PUT@(X,91)-(X+50,99),A%,PSET,0:GOTO 240
260 SOUND7,247:SOUND8,16:SOUND6,10:SOUND12,9:SOUND11,96:SOUND13,9:LINE(90,160)-(
320,100),PSET,2:LINE(549,160)-(320,100),PSET,2
270 IF X=191 OR X=292 OR X=293 OR X=294 OR X=295 OR X=296 OR X=297 OR X=298 OR X
=299 OR X=299 THEN COLOR=(1,1):COLOR=(2,2):GOTO 280 ELSE 310
280 SOUND6,15:SOUND7,7:SOUND8,16:SOUND9,16:SOUND10,16:SOUND12,16:SOUND13,0:SCREE
N7,7:PUT@(X,91)-(X+50,99),A%,PRESET:FOR U=1 TO 20:COLOR=(1,2):COLOR=(1,1):NEXT U
:SCREEN2,7
290 S=S+V*10:LOCATE10,0:COLOR2:PRINTUSING"####";S
300 LINE(90,160)-(320,100),PRESET:LINE(549,160)-(320,100),PRESET:GOTO 200
310 LINE(90,160)-(320,100),PRESET:LINE(549,160)-(320,100),PRESET:GOTO 210
320 PLAY"O3CL8CL8CL8DL4FL8BL2"
330 SCREEN7,7:SYMBOL(70,40),"TIME OVER",5,3,4:SYMBOL(71,41),"TIME OVER",5,3,2
340 SYMBOL(40,100),"PUSH ANY KEY ,AND GAME START",2,2,2:SYMBOL(41,101),"PUSH ANY
 KEY ,AND GAME START",2,2,6
350 IF INKEY$="" THEN 350 ELSE 20
```



◀迷路ゲーム

迷路ゲーム

小塩　誠次（東京都葛飾区）

for FM-7,8,11

オーソドックスな迷路ゲームですが，迷路の大きさを指定できます。また，でき上がった迷路のハードコピーをとったり，自分で迷路を解くこともできます。

RUNすると迷路の大きさを尋ねてくるので，指示された範囲の数字を入力してください。ただし，ディスクBASIC上ではメモリの都合で指示された範囲であっても，大きな数だとOut of Memory が出ることがあるので，そのときは値を入力し直してください。

迷路を描き終わるとハードコピーをとるか尋ねてくるので，とる場合はYまたはyを入力します。次に迷路を画面上で解くかどうか尋ねてくるので，解かない場合はNまたはnを入力すると終了します。それ以外のキーは解くと見なされゲーム開始です。[2]，[4]，[6]，[8]のキーがそれぞれ下，左，右，上に対応して，カーソルが迷路中を移動します。また，エスケープキーを押すと，最短コースを表示してくれるので，それを参考にして自分で解いてください。

〔編集部記〕

迷路プログラムは数多くありますが，最短コースを直ちに示してくれるのは初めてです。穴掘り型の迷路の作り方なので，迷路を作りながら最短コースを見つけられるようです。

FM-7，8用BASICチェックサム（□はFM-11用）

LINE	+0	+1	+2	+3	+4	+5	+6	+7	SUM
10	[65] (FM-11: 8F)	B8	24	70	98	98	A3	05	:[89] (FM-11: B3)
90	F6	FD	77	B0	2F	96	61	B6	:F6
170	65	66	F5	A8	[BA] (FM-11: CA)	1E	09	85	:[CE] (FM-11: DE)
250	52	C5	90	F3	69	CF	8C	86	:E4
330	D1	00	00	00	00	00	00	00	:D1
	[E3] (FM-11: 0D)	E0	20	BB	[EA] (FM-11: FA)	1B	99	C6	/[02] (FM-11: 3C)

●迷路ゲーム●

FM-7，8用（FM-11は変更行を直してください）

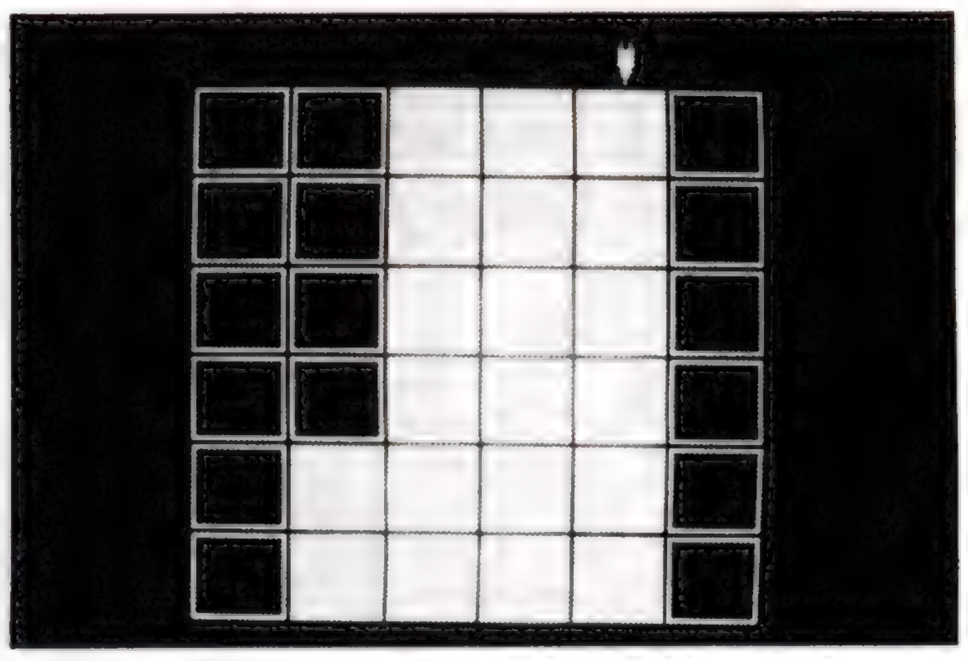
©1984 S.KOSHIO

```
10   DEFINT A-Z:CLS:WIDTH 80,25:RANDOMIZE TIME
20   LOCATE 15,10:BEEP:INPUT "MAZE SIZE 5-47 ";X:IF X<5 OR X>47 THEN 20
30   DIM M(X+1,X+1),X(X*X),Y(X*X),XS(X*X),YS(X*X):N=0:N0=0:Y=X:SY=167/X:SX=SY*2
40   CLS:FOR I=0 TO X+1 STEP X+1:FOR J=1 TO X:M(I,J)=-1:M(J,I)=-1:NEXT J,I
50   LINE(2,8)-(X*SX+3,Y*SY+8),PSET,7,BF:LINE(0,7)-(X*SX+5,Y*SY+9),PSET,1,B
60   SYMBOL(0,0),"START",1,1,4:SYMBOL(X*SX+9,Y*SY+2),"GOAL",1,1,2
70   IF N0=0 THEN XS(N)=X:YS(N)=Y
80   X(N)=X:Y(N)=Y:M(X,Y)=-1:IF X=1 AND Y=1 THEN N0=N
90   LINE((X-1)*SX-XD+4,(Y-1)*SY-YD+9)-(X*SX-XE+2,Y*SY-YE+7),PSET,0,BF:N=N+1
100  R=INT(RND*4):XD=-(R=0):XE=(R=1):YD=-(R=2):YE=(R=3)
110  X=X(N-1)+XD+XE:Y=Y(N-1)+YD+YE:IF M(X,Y)=0 THEN 70
120  FOR I=0 TO 3:IF M(X(N-1)-(I=0)+(I=1),Y(N-1)-(I=2)+(I=3))=0 THEN 100
130  NEXT:IF N>1 THEN N=N-1 :GOTO 100
140  YM=96:A$="HARDC Y/N ?":R=2:C=6:GOSUB 320:BEEP:P$=INPUT$(1)
150  IF P$="Y" OR P$="y" THEN LINE(440,96)-(616,104),PSET,0,BF:HARDC 2
160  T$="00:00:00":B$=T$:YM=106:A$="START Y/N ?":GOSUB 320:BEEP
170  P$=INPUT$(1):IF P$="N" OR P$="n" THEN COLOR 7:END
180  ON INTERVAL GOSUB330:INTERVAL1:TIME$=T$:INTERVAL ON
190  LINE(440,72)-(630,112),PSET,0,BF
200  YM=72:A$="BEST TIME:"+B$:R=1:C=5:GOSUB 320
210  YM=80:A$="YOUR TIME:"+TIME$:C=4:GOSUB 320
220  X=XS(0):Y=YS(0):C=2:GOSUB 310:X=1:Y=1:P=-1
230  X=X-XD*(P=0):Y=Y-YD*(P=0):C=4:GOSUB 310:IF X=2*XS(0)-Y THEN GOSUB 270
240  K$=INKEY$:IF K$=CHR$(&H1B) THEN N=N0:GOSUB 290
250  XD=(K$="4")-(K$="6"):YD=(K$="8")-(K$="2"):C=0:IF K$="" THEN 240
260  P=POINT((X-.5)*SX+XD*SX/2+3,(Y-.5)*SY+YD*SY/2+8):GOSUB 310:GOTO 230
270  YM=96:A$="GAME OVER !!":R=2:C=6:IF B$=T$ OR TIME$<B$ THEN B$=TIME$
280  YM=106:A$="REPLAY Y/N ?":GOSUB 320:INTERVAL OFF:RETURN 170
290  C=0:GOSUB 310:X=XS(N):Y=YS(N):C=2:GOSUB 310:N=N-1:IF N=>0 THEN 290
300  BEEP:RETURN180
310  LINE((X-1)*SX+4,(Y-1)*SY+9)-(X*SX+2,Y*SY+7),PSET,C,BF: RETURN
320  SYMBOL(440,YM),A$,R,1,C:RETURN
330  COLOR 4:LOCATE 65,10:PRINT TIME$:RETURN
```

FM-11用変更行

```
10   SCREEN 0:DEFINT A-Z:CLS:WIDTH 80,25:RANDOMIZE TIME
210  LOCATE 55,10:PRINT CHR$(&H11)+CHR$(4)+"YOUR TIME:"+TIME$
```

縦積み四目並べ“HAMLET”▶



縦積み四目並べ“HAMLET”

—YBK—(福井県福井市)

for FM-7,8,11

このゲームは四目並べです。6×6 のマスの上に交互に石（自分の石は黄色、FMの石は水色)を落として，先に縦，横，斜めのいずれかに4つ石が並んだ方が勝ちです。ただし，石は下から積んでいくので一番下の列か，下に石が積まれたマス以外に石を置くことはできません。

RUN すると，先攻にするかどうか尋ねてくるので先攻ならYまたはyを入力します。それ以外は後攻となります。キー操作は←→キーで石を落としたい列に♥マークを移動させ，↓，または↑キーで落とします。

〔編集部記〕

わずか45行のプログラムでコンピュータと対戦できるとは！　ちなみに，私は1回も勝っておりません。コンピュータの思考時間が長いのは，しかたのない所でしょう。

FM-7, 8用BASICチェックサム（□はFM-11用）

```
               06                      BE       99
LINE  +0  +1/  +2  +3  +4  +5/  +6  +7 \SUM
  10  00 [BC]  92  B4  E9 [71]  00  A6  :[02]
  18  CE  A4   96  1D  2F  00   84  6D  :45
  26  91  92   43  09  00  28   BB  00  :52
  34  4C  9A   00  A1  12  6B   54  7C  :D4
  42  29  AA   00  A4  5B  00   00  00  :D2
-----------------------------------------------
      D4 [36]  6B  1F  85 [04]  93  8F  /[3F]
          80                51            D6
```

●縦積み四目並べ“HAMLET”●

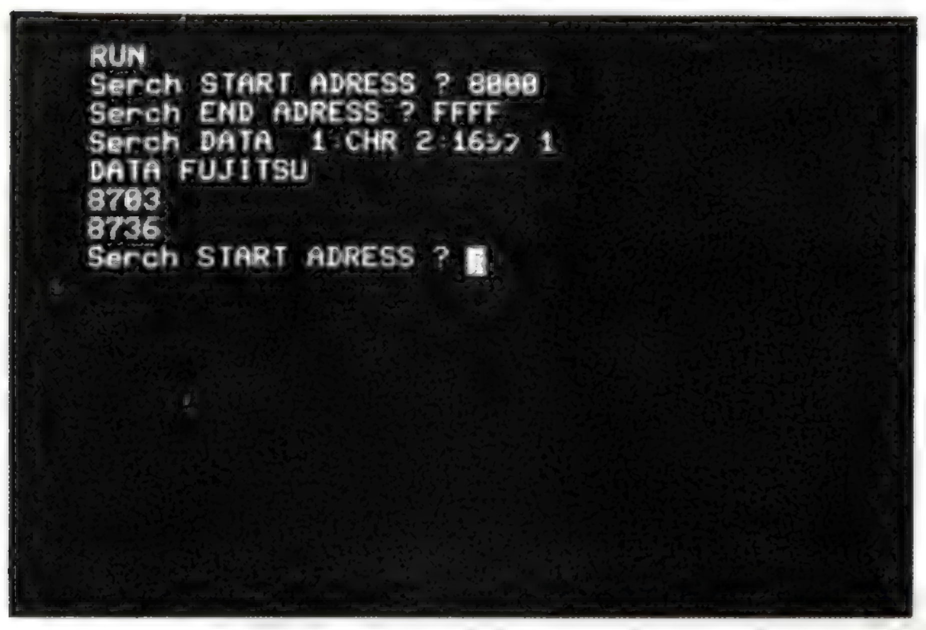
FM-7, 8用（FM-11は変更行を直してください）

©1984 YBK

```
10 ' To be or not to be,that is the question.   [ HAMLET ]   1984,7,1   - YBK -
11 WIDTH80,25:CLEAR:DEFINT A-Y:DIM P(16):RANDOMIZE(TIME/7)
12 FOR X=1 TO 15:READ P(X):NEXT:R=1:Q=2:S=2:Z=3.1415926#/4
13 FOR X= 50 TO 450 STEP 80:FOR Y= 20 TO 170 STEP 30
14 LINE (X,Y)-(X+73,Y+27),PSET,7,B:NEXT:NEXT:V=INT(RND*6)+2:'V=4
15 BEEP:LOCATE 8,1:INPUT "SENKOU (Y)";A$:IF A$<>"Y" AND A$<>"y" THEN 28
16 '
17 M=80*L+82:LINE (M-78,4)-(M+95,17),PSET,0,BF:SYMBOL(M,4),"♥",2,2,2
18 A$=INKEY$:A=ASC(A$+" "):IF (A<28)+(A>31) THEN 18
19 IF A>29 THEN IF N(L+2)<6 THEN 21 ELSE BEEP:GOTO 18
20 L=L+(L>0)*SGN(A-28)+(L<5)*SGN(A-29):GOTO 17
21 BEEP 1:S=L+2:T=N(S)+2:N(S)=T-1:M(S,T)=1:C=6:GOSUB 32
22 GOSUB 37:IF P>5000 THEN MES$="ｱﾅﾀﾉ ｶﾁ !":GOTO 46
23 '
24 GOSUB 31:U=-30000:FOR S=2 TO 7:T=N(S)+2:IF T>7 THEN NEXT:GOTO 28
25 GOSUB 35:W=P:IF P>5000 THEN F=1:V=S:GOTO28 ELSE GOSUB 35:W=W+1.1*P
26 IF T<7 THEN GOSUB 34:W=W-.5*P:GOSUB 34:W=W-.6*P ELSE W=W-40
27 IF W>U THEN U=W:V=S:NEXT ELSE NEXT
28 S=V:T=N(S)+2:N(S)=T-1:M(S,T)=2:C=5:GOSUB 32
29 IF F<>1 THEN GOSUB 31:BEEP:GOTO 17 ELSE MES$="ﾜﾀｼﾉ ｶﾁ !":GOTO 46
30 '
31 N=N+1:IF N<>36 THEN RETURN ELSE MES$="ﾋｷﾜｹ ﾃﾞｽﾈ !":GOTO 46
32 BEEP 0:PAINT(80*S-109,231-30*T),C,7:RETURN
33 '
34 M(S,T)=R:T=T+1:GOSUB 35:T=T-1:M(S,T)=0:RETURN
35 SWAP R,Q:M(S,T)=R:GOSUB 37:M(S,T)=0:RETURN
36 '
37 P=0:FOR H=0 TO 3:D=SGN(COS(H*Z)):E=SGN(SIN(H*Z)):X=S:Y=T
38 IF X>=2 AND X<8 AND Y>=2 THEN X=X-D:Y=Y-E:GOTO 38
39 FOR J=0 TO 2:B=5+J:IF X+B*D<1 OR X+B*D>8 OR Y+B*E>8 THEN 43
40 IF M(X+J*D,Y+J*E)=R OR M(X+B*D,Y+B*E)=R THEN NEXT J:GOTO 43
41 G=0:FOR K=J+1 TO J+4:IF M(X+K*D,Y+K*E)=Q THEN NEXT J:GOTO 43
42 G=2*G-(M(X+K*D,Y+K*E)=R):NEXT K:P=P+P(G):NEXT J
43 NEXT H:P=P-10*ABS(S-4.4)-15*ABS(T-4.6):RETURN
44 '
45 DATA 0,0,40,0,20,60,200,0,10,20,200,40,200,200,7000
46 CLS 1:PRINT MES$:IN$=INPUT$(1):CONSOLE 0,25
```

FM-11用変更行

```
11 SCREEN 4:WIDTH80,25:CLS:CLEAR:DEFINT A-Y:DIM P(16):RANDOMIZE(TIME/7)
15 BEEP:LOCATE 8,1:INPUT "SENKOU (Y)";A$:CLS 1:IF A$<>"Y" AND A$<>"y" THEN 28
```



◀ DATA SEARCHER

DATA SEARCHER

成瀬　健一（東京都国分寺市）

for FM-7,8

メモリ上のデータの中から，任意の文字列，または16進データ列をサーチし，その先頭アドレスを表示します。ただし，サーチできる長さは文字列は255文字，16進データは127バイトまでです。機械語を使用しているので64KBを1秒弱でサーチします。

使い方は，RUNするとサーチを始めたいアドレスを尋ねてくるので，&Hをつけないで16進を入力します。次に終了アドレスも同様に入力します。今度はサーチしたいデータの属性の入力です。[1]でキャラクタサーチ，[2]で16進サーチになります。属性の次にサーチしたいデータを入力します。[1]の場合はそのまま入力，16進の場合は&Hを付けたり，カンマで区切ったりしないで続けて16進数を入力してください。上位4ビットが0の場合は0も入力します。

データが見つかると画面に表示しBEEP音で終了します。BEEP音だけで，何も表示しないときは，探したいデータが見つからなかった場合です。

〔編集部記〕

機械語なので速いのは当然なんですけど，やっぱり速いですね。でも，POKE文でオペランドを直接つけ換えるのは，本当はいけないんですよ。なお，機械語部分は位置独立なので1000行の「M」の値とCLEAR文のオペランドを適当に変更してください。プログラムの使うアドレスはM=&H5000のとき$5000から$512Fまでで，メインプログラムの後ろに256バイトのワークエリアを含んでいます。

FM-7用BASICチェックサム（□はFM-8用）

LINE	+0	+1	+2	+3	+4	+5	+6	+7	SUM
1000	FE	47	9E	3F	FC	96	66	38	:52
1080	0A	75	EE	B8	6F	FE	B4	72	:B8
1160	FE	A0	C4	D0	B4	06 [26]	DC [DD]	00	:C8 [E9]
	06	5C	50	C7	1F	9A [BA]	F6 [F7]	AA	/D2 [F3]

●DATA SEARCHER●

FM-7用（FM-8は変更行を直してください。）

©1984 K.NARUSE

```
1000 CLEAR,&H4FFF:M=&H5000
1010 FOR P=M TO M+47
1020 READ D$:POKE P,VAL("&H"+D$)
1030 NEXT
1040 INPUT"Serch START ADRESS ";A$:IF A$="" THEN A=&H8000 ELSE A=VAL("&H"+A$)
1050 INPUT"Serch END ADRESS ";B$:IF B$="" THEN B=&HFBFF ELSE B=VAL("&H"+B$)
1060 INPUT"Serch DATA  1:CHR 2:16ｼﾝ ",E:IF E<>1 AND E<>2 THEN BEEP:GOTO 1060
1070 LINEINPUT"DATA ";C$:IF C$="" THEN 1070 ELSE C=LEN(C$)
1080 POKE M+1,INT(A/256):POKE M+2,A-INT(A/256)*256
1090 POKE M+41,INT(B/256):POKE M+42,B-INT(B/256)*256
1100 POKE M+13,C/E
1110 Q=M+48:FOR P=1 TO C STEP E
1120 IF E=1 THEN R=ASC(MID$(C$,P)) ELSE R=VAL("&H"+MID$(C$,P,2))
1130 POKE Q,R
1140 Q=Q+1:NEXT
1150 EXEC M
1160 GOTO 1040
1170 DATA 8E,00,00,30,1F,30,01,1F
1180 DATA 12,CE,50,30,C6,00,A6,C0
1190 DATA A1,A0,26,14,5A,26,F7,1F
1200 DATA 10,30,01,FE,02,17,AF,41
1210 DATA BD,AC,37,BD,9B,50,20,D8
1220 DATA 8C,00,00,26,D8,7E,DB,49
```

FM-8用変更行

```
1210 DATA BD,AF,E3,BD,9F,97,20,D8
1220 DATA 8C,00,00,26,D8,7E,DD,48
```

ショートプログラム応募要領

■1行80字以内で，できるだけ30行に収めてください。
■きれいなプログラムを望みます。マルチステートメントは多用しないでください。
■プログラムは，カセットテープまたはディスケットに入れてお送りください。ディスケットの場合は，新しいディスケットをお返し致します。
■プログラムの説明，使い方など必要なことを簡潔にまとめてください。
■プログラムおよび原稿は，多少修正させていただく場合があります。
■他誌との二重投稿，盗作などモラルに反する応募は固くお断わり致します。
■応募原稿は原則としてお返し致しません。
■応募された方全員にOh！FMシールとシャープペンシルをお送り致します。
■すぐれた作品は本誌に掲載し，原稿料にかえてOh！FM特製トレーナと図書券5,000円分をお送り致します。
■原稿にも，住所・氏名・年齢・職業などを明記して，〒102 東京都千代田区四番町2－1 (株)日本ソフトバンクOh！FMショートプログラム集係までお送りください。

好評発売中

あなたも今日から
プログラマー!!

BEST ベスト

FOR BUSINESS EVERY PURPOSE SOFTWARE TOOL

ビジネスユースの為の
簡易ソフトウェアツール!!

適応機種

- 富士通FM-7
- 富士通FM-8
- 富士通FM-11
- PC-8800シリーズ
- PC-9800シリーズ

特　　徴

1. 難しいプログラムをパーツ化!

- 難しいフログラムの部分をパーツ化してフロッヒーに格納してありますからお客様は自由に取り出して、つなぎあわせ、必要な変数やデータを書き込むだけで、本格的なビジネスフログラムが作成できます。
- ハッケージソフトや、汎用ソフトと違い、内容は全くのハンドメイドですから利用者の思想が充分盛り込め、拡張性があります。

提供媒体　●フロッヒィディスク（5インチ・8インチ）

※上記に関するお問い合わせは企画部 福岡・柘植まで.

日本コンピュータスクール事業協同組合加盟

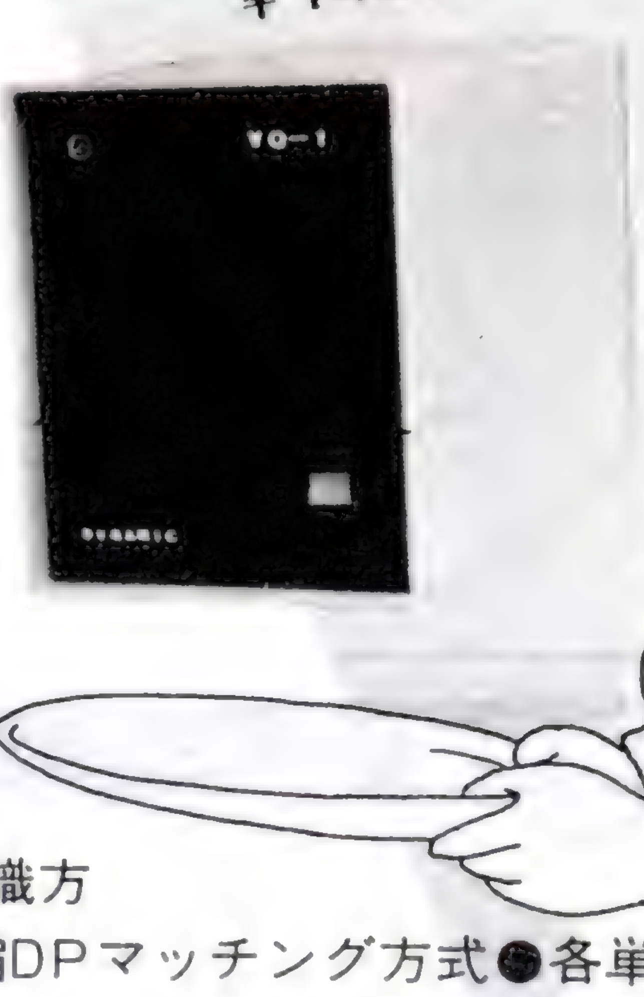
理工産業株式会社

太陽グループ　〒510　三重県四日市市九の城町４-２０
TEL〈0593〉51－1651㈹ FAX0593－51－5104

- 四日市 〒510 三重県四日市市九の城町4-20 ☎(0593)51-1651
- 津 〒514 三重県津市本町23-17 ☎(0592)25-6251
- 松阪 〒515 三重県松阪市船江町785 ☎(0598)51-4312 FAX 51－0602

音声認識装置

パーソナルコンピュータ用



型式VO-01
¥148,000

★特徴

音声をあらかじめメモリにパターン化して登録しておき入力された音声のパターンと比較照合しその結果を出力します。パソコンの指示によりパターンの変更も可能です。一度登録されたパターンは、パソコン側のディスクに記録でき再登録の必要はありません。

★機能

特定話者、離散発声単語認識方式●登録単語数128語●圧縮DPマッチング方式●各単語音声の長さ最大2秒●平均応答時間0.5秒●認識率98%以上

※パーソナルコンピュータとの接続はRS232-Cポートを用います。

OAソフト—03　顧客管理プログラム
日本語処理により最大12項目データ件数3,000件迄可能　¥38,000

OAソフト—01　プログラムレス ソフト
各種帳表作成のためのデータ入力、計算、ソート、検索を自在に応用できるプログラムです。　¥76,000

OAソフト—04 見積書発行プログラム
日本語処理による各種見積作成プログラム書式については要指定　近日発売予定

★音声認識装置応用による各種プログラムを制を制作いたします。

ダイナミック株式会社

TEL0476(24)2423代表　千葉県成田市東和田355番地

※バックナンバーは、取扱い店へご注文になるか、直接、鈴木技研へお申し込みください。

▶全国有名書店及びマイコンショップでお申し込み、お買い求めください。
▶お電話によるお問い合わせは、月～金曜日までのAM11:00～PM4:00までとさせて頂きます。

■お申し込み・お問い合わせは――
鈴木技研 〒354 埼玉県富士見市三芳郵便局私書箱28号
☎0492(52)2513

ソフトマートから感激のソフト新登場!!

日本語FLEX
(FM-7／8用)

FLEXに漢字機能を搭載した新FLEX。その名も「日本語FLEX」。FM-7/8用として新登場です。
すべてのアプリケーションで日本語が使用でき、またほとんどの言語で漢字が扱えます。

スクリーンエディット搭載

F-BASICと同等のスクリーンエディトが、英数カナモード時にできます。もちろんEDIT時に限られたわけではありません。これにより開発時間は大幅に短縮されます。

ミニワープロとしても使用可能

その上、基本的なワープロとしての機能が組み込まれています。プリンタに対して「マージン設定、文字位置指定、行のスキップ、文字間隔指定、文字サイズ指定」などができ、また付属の「GAIJI」コマンドを使うことにより、外字登録が可能です。

ユーティリティコマンドも増加

ユーティリティコマンドも下記のように多く追加されています。

- ●「COLOR」 ： 画面の色を変える。
- ●「CONSOLE」 ： コンソール関係を変える。
- ●「GAIJI」 ： 外字登録をする。
- ●「JFRMAT」 ： 文書清書をする。
- ●「JLIST」 ： 日本語混じりのプログラムをプリントする。
- ●「KEYLIST」 ： ファンクションキーに登録されている文字列を表示する。
- ●「PALETTE」 ： パレット機能を変える。(FM-7のみ)
- ●「PRINTER」 ： プリンタの印字モード（英数カナモード、日本語モード）を変える。
- ●「SETPF」 ： ファンクションキーに文字列を設定する。(アセンブラ、XBASIC、C、PASCALなどは簡単に設定できるようになっている)。
- ●「SUBMIT」 ： 擬似的にパラメータを使用できるようにする。
- ●「TAB」 ： TABステップを設定する。(C、PASCAL、BASIC、アセンブラ用のTAB位置は簡単に定義できるようになっている。)
- ●「TIME」 ： 内部クロックの時間を調べたり設定したりする。

(JFORMATと、JLISTはC言語で記述したソースプログラムも付いています。)

ユーザーRAMが変らない

以上のように多くのものをサポートしているにもかかわらず、$0000～$BFFFまではユーザーエリアとして開放されています。したがって今までのFLEXで開発していた方でも日本語FLEXを同じように使えます。

価　　格

■日本語FLEX for FM-7/8……………………¥63,000

(日本語FLEXへのバージョンアップは¥20,000で行ないます。詳しくはお問い合わせ下さい。)

※ FLEX はTSC社の登録商標です。

SMart SOFT MART, inc.

TSC社日本総代理店・スーパーソフト社 JAPAN AUTHORIZED DISTRIBUTOR

ソフトマート株式会社

〒101 東京都千代田区神田須田町1-18-6 第1谷ビル ☎(03)256-5881 FAX(03)256-6180 GII/GIII

Nashua
Professional Magnetic Media
遂に上陸。幻のNASHUA
USAで高品質と評判の高いNashuaがとうとう日本にやって来た。
そこでモニター募集
両面倍密2枚1組を先着50名様にプレゼント。ハガキに下部のモニター応募券を貼り、住所、氏名、年齢、電話番号、職業をご記入のうえ下記まで。
販売代理店募集中
資料のご請求、および当社販売代理店のお問い合せは、ハガキか電話で下記まで。
輸入発売元 (株)コスモ製作所
〒180 東京都武蔵野市吉祥寺本町2-2-2
☎0422(20)0450代

インテックFM・ユーティリティソフトシリーズ

HOT INFORMATION

中古パソコン情報サービス…インテック

FMファン待望の

ディスクアナライザー登場「Mr.ノーティ」

FM-7/8/NEW7/11用

FM唯一のREAD、WRITEもできる
本格派ディスク解析ツール。

定価 ¥6,800

●申し込み方法TEL、またははがきでお申込み下さい。

FMにべんりの

べんり君FM

FM-7用

① FILE COPY(特定ファイル転送)

このプログラムは、マシン語、データ文、BASICアスキーファイルなどすべてのファイルをあつかえ、あなたが選んだ特定のファイルを自由に転送できます。

②SKEW FORMAT (スキュー・フォーマット)

ディスクにスキューをかけてフォーマットするプログラムでありスキューをかけていない時に比べて、LOAD,SAVE にかかる時間が数分の1に、向上します。また、他の類似品に比べて、スキューファクタに偶数をとることができるというのも大きな特長です。

③SKEW COPY (スキューディスク高速コピー)

富士通システム・ディスクの中にある「VOLCOPY」 でスキューのかかったデスクのコピーを行うと大変時間がかかるのでスキューがかかっているディスクどうし、又は片方のみスキューがかかっているディスクのコピーをスキューのかかっていない状態のコピーと同速で行なう為のプログラムです。

④WRITE FILES (DIR4列表示)

"FILES" 命令によるファイルの出力を横方向にすることにより、ディスク内のファイルの格納状態に対し明瞭性と全体把握性が格段に向上し類似プログラムに比べてDISK BASICを直接書換えるので、一旦書換えが終了したならば、後はRESETしてもずっとFILES が横に表示される。ファイルネームにSFコマンドやRCコマンドがあってもオーダ動作の2点がすぐれています。

⑤DISK LOADER (ディスク内マシン語アドレス一発表示)

ディスク内に存在するファイルの中でマシン語ファイルの開始番地終了番地、実行番地を調べることを目的としたプログラムでディスクにSAVEM したマシン語のアドレスをいちいち紙に書いて記録しておく必要がなくなり他の類似フログラムに比べ調査速度が速いこと、すべてのマシン語ファイルを調べること、そして、最っとも大切なことは実行番地の調査が正確であること、の3点がすぐれています。

定価 ¥6,800

●支払い方法商品到着後、動作確認のうえ下記の口座にお振込みください。
●住友銀行(普)下高井戸支店156709

inn-tec.

〒168 東京都杉並区下高井戸1-7-7-315

TEL. (03)324-9764

FM-11 F9450-II ユーザーにPFD-8の良さを！

5吋1MBを実装した両機には同一ソフトで動作する8吋も便利です

FM-11AD2/BS

汎用パソコンとして、最強の機能を誇る名機。OS-9レベル2を標準装備し、マルチタスク、マルチウインドウ、リアルタイム処理が可能。
1MBまでのRAMをサポートした、BASIC09も添付されています。(AD2モデル)
このFM-11AD2/BSに、8インチフロッピィディスクの名機、PFD-8Fが接続します。

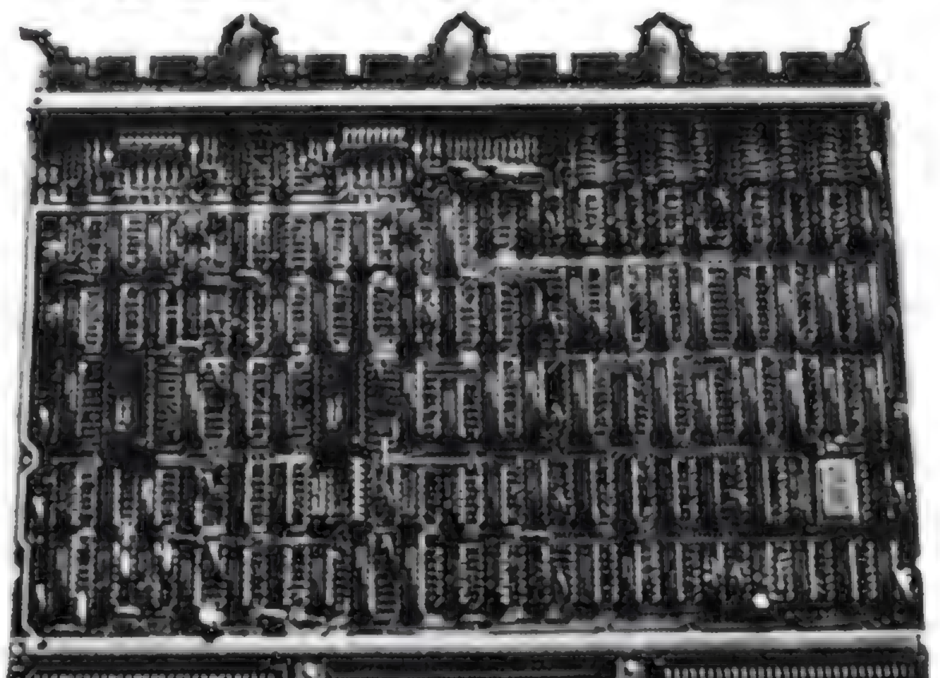
新製品：FM-77にも接続可能！
MB-22454インタフェースボードが必要です。

FACOM9450-II

16ビット・パソコンの名機です。同時に2役のマルチ・ジョブ機能でひときわ有名になりました。
文字通り、実戦向きのOAパソコンでしょう。
(パナファコムC-280、松下C-7000も同等です)
このFACOM9450-IIに、8インチフロッピィディスクの名機、PFD-8Fが接続します。
F9450-II用ケーブル別売　¥10,000

F9450-II専用ボードシリーズ
- ユニバーサルボード　¥24,000
- RS-232C+DI/DOボード¥150,000

PIO-1022 汎用インタフェースアダプタ　¥150,000



F9450-II パソコン専用のインタフエースアダプタで、パソコン本体の空スロットに実装するボードです。
純正には、ない組合せを1枚のボードに実現しています。
RS-232C×2CHと、デジタル入出力×2ワードの仕様を1枚のボードに組込んでいます。ソフトウェア的には、純正品のものと完全コンパチブルですから、安心して利用できます。

PFD-8F 8インチフロッピィディスクユニット

(F9450-II、C-280、C-7000では、別売ケーブルが必要です。)

- 8インチ両面倍密度ドライブ2台装備、2MB容量。
- 超静寂冷却ファン採用。ドライブ番号設定可能。
- サービスコンセント2個付。
- 接続ケーブル添付。

(FM-11ST/AD/EXでは、MB22415が必要です)
(FM-77では、別売インタフェースボードが必要です)

¥239,000

お急ぎの方、代金引換にて直販も致します。御連絡下さい。

I·O DATA 情報と制御のシステムメーカー
株式会社 I·Oデータ機器
本社：金沢市高岡町7-22 ☎0762-21-4812
東京営業所：☎03-258-2040
●代理店
関東電子(株) ☎03-257-6221
(株)ニデコ ☎03-251-8061
近畿システムサービス ☎06-644-6625
㈱アスターインターナショナル ☎03-257-0128

通信販売のお申し込みは
03-251-9911
全セット先着8名様に限り特典を用意いたしております。

東京 名古屋 札幌 ツクモ 全店 8号店 8/15〜9/20
オープン記念大特価セール

FM-NEW7 セット 新製品

ニューセブン

本体価格 (MB25015) ¥99,800

★只今、FM-NEW7セットをお買上げの方にデータレコーダをプレゼント!!

8号店開店につき、各セットにそれぞれサービス品を用意致しております。

FM-NEW7 ¥99,800	FM-NEW7 ¥99,800	FM-NEW7 ¥99,800
スペシャルセット	Bセット	Cセット
U-0021 富士通MB27307同等品12″ D.P.O.31、2000文字 モニターケーブル ツクモオリジナル ソフト5本 「RAM DISK」	CV14-VC1(サンヨー) 14インチ・パソコンモニターテレビ、2000文字対応 モニターケーブル ツクモオリジナル ソフト5本 「RAM DISK」ソフト	TH14NM2G ナショナル14インチ1000文字 または TX-12TI ナショナル12インチ1000文字 モニターケーブル ツクモオリジナル ソフト5本 「RAM DISK」ソフト
	定価¥112,500	定価¥84,300
特価¥65,000	特価¥65,000	特価¥39,000
合計特価¥159,800	合計特価¥159,800	合計特価¥133,800

ツクモメンバーズ
Z募集中デス!
おハガキでお申し込み下さい! すぐ申込用紙をお送りしま〜す
〒101 東京都千代田区外神田3-2-4 東神ビル1F (株)九十九電機 ツクモメンバーズ事務局
入会金無料!

特注ビジネスソフト
Zのごあんない

パソコン用ビジネスソフトの特別注文に応じるスタッフを御紹介致します ソフト勉強会や新技術利用のアドバイスも受けられます。モチロン、オフコン、ミニコンのオーダーソフトより低価格デス!
詳しくは ☎03-251-9966 九十九電機ソフトセンター まで

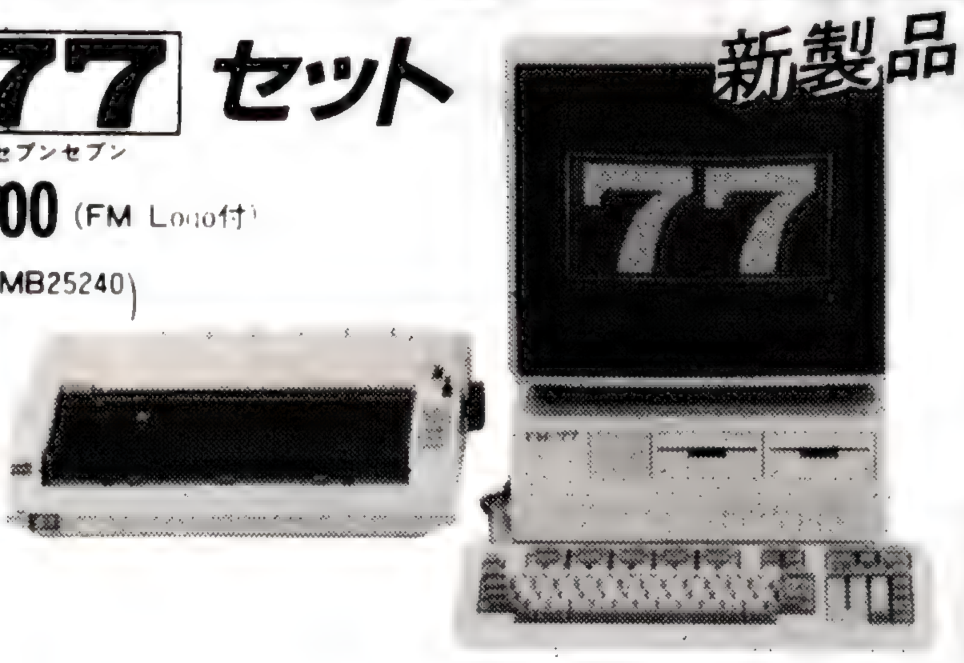
FM-77D2 ¥228,000	FM-77D2 ¥228,000
スペシャルセット	Bセット
MB-27307同等品 (U-0021) モニターケーブル 3.5インチ版 ゲームパック	MB-27307同等品 (ナショナルTX-12M2) モニターケーブル 3.5インチ版 ゲームパック
定価¥156,800	定価¥101,600
特価¥65,000	特価¥50,000
合計特価¥281,000	合計特価¥266,800

★只今、FM-77D2セットをお買上げの方に3.5インチディスケット10枚・3.5インチフロッピーケースをプレゼント!!

FM-77 セット 新製品

セブンセブン

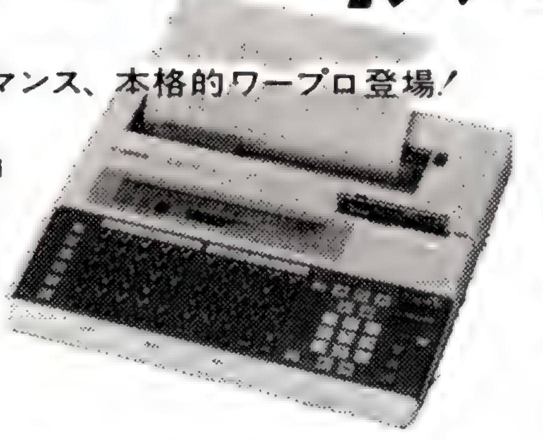
本体価格 (MB25250) ¥228,000 (FM Logo付)
(1ディスクドライブタイプ(D1)MB25240 ¥198,000もあります。)



ワープロご購入のチャンス!!
ツクモ全店 限定8台 大特価セール (店頭販売のみ)
特価品の内容は直接各店へお問い合わせ下さい
ツクモ8号店開店記念

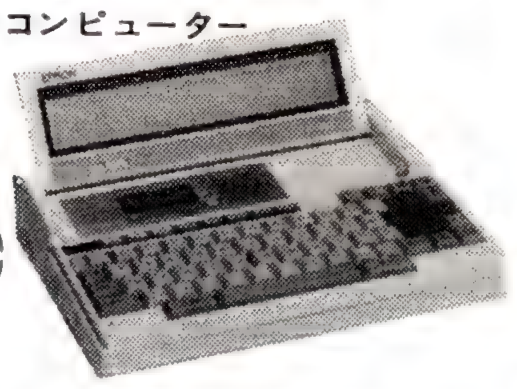
ツクモ特選 秋のワープロ特集!!

ハイコストパフォーマンス、本格的ワープロ登場!
CANON
パーソナルワープロ
PW-10
¥148,000



★いつでも、どこでも自由自在! B4サイズ、3kgの手軽さ ★50音、たて配列、大型液晶ディスプレイ、送り仮名判断機能etc! ★辞書は42,190語、文字種3,956文字、外部記憶はメモリーカード方式と、パーソナルでも機能は本格派 ★楽しい3色、ブラック・ホワイト・レッド。お好みのカラーが選べます。

ハンドヘルドで初めて、日本語ワープロ標準装備
新入力方式タッチ16採用
日本語ハンドヘルドコンピューター
EPSON
HC-88
(タッチ16、JIS)
(JISタイプはタッチ16としても使用可)



HC-88	タッチ16キーボードタイプ	¥298,000
HC-88	JISキーボードタイプ	¥298,000
HC-80		¥198,000

ワープロ第2世代
富士通
OASYS Lite
〈オアシス・ライト〉
¥220,000

★日本語を弾くように打つ、親指シフトキーボード
★日本語はもちろん、欧文もOKの国際派
★編集作業も自由自在
★鮮やかな印刷、熱転写プリンタ採用

いつでも、どこでも、誰でも打てる初めての日本語パーソナルライター
ブラザー
ピコワード
¥88,000

★誰でも打てるカンタン操作、便利な訂正機能、はじめてでもすぐわかる50音順配列、豊富な文字種(約3500種)、印字のきれいな熱転写プリンター

マイコンワープロセット

● 漢字ROM＋セットサービス品データレコーダー＋MB27303同等品TX-12M2(ナショナル12インチモニター)＋モニターケーブル＋MB27406同等品(ナショナルプリンター)＋プリンターケーブル＋用紙(500枚)＋ワープロソフト「文太」TAPE版
＝ワープロ用セット定価¥340,700➡セット特価¥143,200

FM-NEW7(本体)¥99,800＋ワープロ用セット 合計特価¥238,000

コンビニエンスDOS7 新発売

富士通FM77用3.5インチディスク版
¥9,800 送料¥500

プリンターセット

FM-7用プリンターお買い上げの方には「プリンターバッファ(TAPE版ソフト)」プレゼント!!

富士通 MB27406
● FM用 富士通MB27406同等品 プリンター(ナショナル)

限定50台 特価¥49,800
ケーブル、用紙別売(¥4,000)

新発売 FM-8・FM-7でおなじみの
なんでもかんでもユーティリティー

富士通FM77用3.5インチディスク版
¥9,800 送料¥500

通信販売の受付は
03(251)9911へ
ご注文の際は、ご注文になりたい商品名をご記入の上、現金書留にてお願い致します。20万円以上は銀行振込みも取り扱っております。お問い合わせ下さい。
富士銀行 神田支店
普通口座 No.894047

即決 ツクモ全国クレジットOK!!
● 現金特別価格でツクモクレジットが利用できます。残金のみに金利がかかります。
● 60回払いまで出来ます。但し、1回の支払い額は3,000円以上。
● その場で、お持ち帰りできるクレジットもあります。
● 印鑑、身分証明書(免許証など)が必要です。
● 取扱いクレジットカード:日本信販、JCB、DC、UC、VISA、MC、CF、ジャックス
(尚、クレジットカードご使用の場合は店頭員にご相談ください)

■通信販売は 〒101 東京都千代田区外神田郵便局私書箱135号 九十九電機(株) FM-9係

九十九電機株式会社

■5号店(マイコン・ハム・ビデオ)	〒101	東京都千代田区外神田3-1-14	☎03(251)0531〜2
■7号店(マイコン)	〒101	東京都千代田区外神田1-9-9	☎03(253)4199
■8号店(電話器・通信機)	〒101	東京都千代田区外神田3-14-9	☎03(251)0099
■万世店(通信機)	〒101	東京都千代田区外神田1-3-9	☎03(251)2441〜3
■ラジオセンター店(ポケコン)	〒101	東京都千代田区外神田1-14-2	☎03(251)2657
■ニューセンター店(マイコン)	〒101	東京都千代田区外神田1-16-10	☎03(251)0986〜7
■札幌店(ビデオセンター)	〒060	札幌市中央区南二条西4 乙井(おとい)ビル1F・B1	☎011(241)3999
■札幌店(マイコン・ハム)	〒060	札幌市中央区南二条西3-15-1 さっしんビルB1	☎011(241)2299
■名古屋店2F(オーディオ・ビデオ)	〒460	名古屋市中区大須3-30-86	☎052(263)1655〜6
■名古屋店3F(マイコン・ハム)		大須ラジオセンター・アメ横ビル	☎052(263)1681

■ご注文商品の中、未着の品物がございました時は、通販受付電話で納品書No.をお知らせ下さい。尚、商品の詳しい取り扱い等は、各販売店までお問い合わせ下さい。
■定休日 東京各店は毎週木曜日と第3水曜日、名古屋店は毎週月曜日
札幌店(マイコン・ビデオ店)は第3木曜日

★セットの場合は梱包送料¥2,000を加算して下さい。

NEW タテ形コンピュータデスク。

ウエルとオリンパスA.V.S(オリンパス光学工業系)と共同企画による‥‥

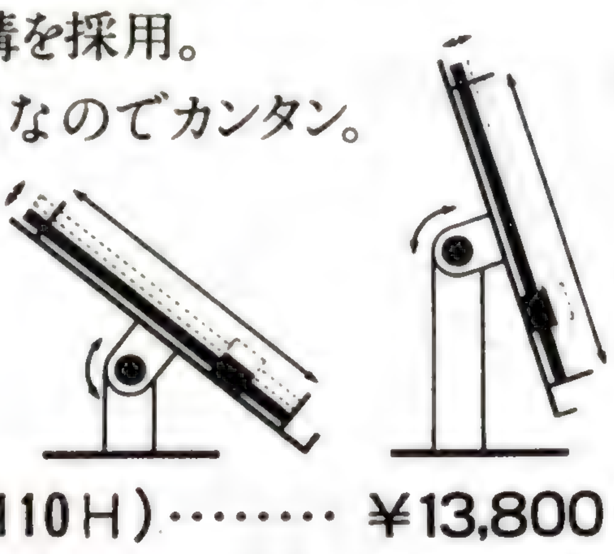
❶スペースを最小限におさえた、タテ形コンピュータデスク。
❷人間工学の研究から生まれたモジュール。
❸インプットの必需品〈原稿スタンド〉を標準装備。
❹キーボードはコード付のまま収納でき、誤操作 ほこりを防ぎます。
❺組みたて式。備付のレンチ一本で女性にも15分で 組みたてOK。
❻基本色はホワイトベージュ、だれにも親しめる美しいデザイン。
❼帯電防止対策用マット、アースを完備。(オプション)
❽ペーパー用トレイ、原稿スタンド用カーソルも用意。(オプション)
❾教育用、展示会用などに持ち運べるキャリングケース。(オプション)
❿多機能と豊富な装備で量産化に成功。お求めやすい価格を実現。

computer system furniture PAT.P
ウエルデスク PW520 (原稿スタンド付) ¥58,800

NEW インプットの必需品。

データスタンド3機種‥‥‥‥‥‥‥‥‥‥‥‥‥‥‥ウエルから新発売。

●入力の作業能率がぐ～んと上がる特殊バネ機構のカーソル。
●オペレータのイライラを解消し入力ミスを防ぎます。
●まとまった原稿は、15㎜厚まで載せてカーソル操作できます。
●原稿面は、タテ・ヨコ両方に可動する新機構を採用。
●原稿の入れ替えは、カーソルが180°回転式なのでカンタン。
●ふらつきのない堅牢なスチール製。
●色はホワイトベージュ。
●用途に合わせて選べる3機種。
●美しいデザイン。



1. データスタンドA-1(原稿面300×300、台110H)‥‥‥‥ ¥13,800
2. データスタンドA-2(原稿面300×300、台195H)‥‥‥‥ ¥14,800
3. データスタンドB-1(原稿面400×300、台110H)‥‥‥‥ ¥15,800

ウエルデータスタンド PAT.P

●コンピュータデスクのベストセラーOA1100。

●製品に関するお問合わせは、直接連絡くださるか、又は右下の請求欄に☑印・ご氏名・ご住所・電話番号を明記の上はがきに貼ってお送りください。カタログおよび購入方法についてお送りします。

●ご注文は電話でもOK。——全国どこへも直送。

☎(03)630-3446

人間とコンピュータの新しい調和。

WELL™

製造発売元
株式会社コミネエージェンシー・ウエル事業部
〒135 東京都江東区冬木22-22 ハイマート201
本社 東京都千代田区三番町3-3スイセイシャビル

□カタログ希望
□注文したい
ご氏名
ご住所 〒 ☎( )
ご職業
ご使用機種又は購入予定機種名

FM-N

創刊

日経バイト
NIKKEI BYTE

まもなく創刊

いまお申込みになれば創刊前・秋号もお読みになれます。

すでに購読を申込まれ、
創刊前・春号、夏号をお読みになられた方々から、
質の高い、読みごたえのある内容と
高く評価されています。

創刊前・秋号を贈呈
(8月20日発行)

あなたもいますぐお申込みください。

わが国の先進的なパソコン・ユーザー6万人の方々が、すでに購読を申込まれています。

クリエイティブなパソコン活用に役立つ、本格的な情報誌と大きな反響を呼び、お申込みが集中。9月26日には、いよいよ創刊号を発送します。

あの「BYTE」誌と全面提携

世界で最も評価の高い米国マグロウヒル社発行のパソコン誌「BYTE」。その記事の中からわが国のパソコン・ユーザーにとって重要な情報を厳選し、毎号翻訳・掲載します(「日経バイト」全誌面の約35%)。

「日経バイト」編集の3つの柱。

●パソコンの最新技術、最新製品を詳しく、わかりやすく解説

高度な専門知識を持つ記者が、新技術、新製品の中から注目すべきものを徹底分析。性能・機能をはじめ、技術・製品の新たな活用法、技術の派生など、一歩踏み込んで解説します。

●高度なパソコン利用に不可欠の基礎技術の解説

OSや言語の内部構造、ニューチップの動作原理など、パソコンの中級・上級者の方が、より高度なパソコン活用をめざすときに不可欠の基礎技術を、幅広く、体系だてて解説します。

●アイディアに富んだ、楽しいホビー情報の提供

アイデアフルなボードやプログラムの自作法、ゲームなどの楽しいホビー情報にも力をいれます。この分野では定評のある米国の情報も、独自のネットワークでいち早く紹介します。

特典 いますぐ購読をお申込みの方へ創刊前・秋号(8月20日発行)を無料贈呈

●10月1日創刊 ●月刊(毎月1日発行)
●A4変型判、毎号約200頁
●個人の年間予約購読、ご自宅郵送制

購読料金

★1年購読(12冊)……………… 6,000円(1冊当たり500円)
★3年購読(36冊)………………12,000円(1冊当たり333円)
料金お支払いは創刊号をお届けしてからで結構です。

発行:日経マグロウヒル社

お申込みは裏面のハガキでいますぐ

創刊 日経バイト NIKKEI BYTE 創刊前・秋号を発行

創刊前に先き立ち、あなたに秋号を贈呈

創刊前・秋号の特集は、注目のパーソナル・データ通信。「BYTE」誌からホビー情報も紹介。

創刊前・秋号の主な内容

【特集】パーソナル・データ通信

新たな可能性を探る

通信回線に接続したコンピュータを介し、メッセージ交換を行う電子掲示板サービスが全米で流行中。パソコン間通信プロトコルは米国ではハードの追加が必要ないMNPプロトコルが有力。信頼性の高いデータ通信が行えるモデム内蔵電話機も出揃う。このパーソナル通信の概要と日本での実現可能性を探る。

【解説】

●RAMディスク・ユーティリティ

最近のパソコンは、比較的大容量の主記憶装置をもつが、必ずしも全部を使っていない。その未使用主記憶領域の1部をディスク装置として扱えるようにするソフトウェアがいくつか登場。その機能、性能の概略を紹介。

【新技術、新製品レビュー】

●プログラム言語としてのdBASE II

dBASE IIの利用者が、リレーショナル・データベースとしての機能面よりも、Basicに代わるプログラム言語としての使い勝手に着目して、dBASE II漢字版を評価。

● Thinktank：ワードプロセサの新顔、"アイデア・プロセサ"「BYTE」誌より

単なるワード・プロセシング・プログラムを越えて、考えを系統だてる機能をもった、ユニークなソフトをとりあげ評価する。

【ホビー情報】発信・着信両用MODEM ECM-103の製作「BYTE」誌より

MODEMの動作原理の解説と、30個の部品で構成できる300ビット／秒タイプの高信頼性MODEMの製作。

【バイトネットワーク】 パソコン分野の新しい出来事をいち早くレポート。

【フォトレビュー】 話題の新製品などを写真で解剖。

【マイクロバイト】 米国最新情報を「BYTE」誌と同時掲載。

【新製品】【クラブニュース】【カレンダー】…ほか。

無料贈呈 いますぐ購読をお申込みの方へこの秋号を贈呈。

この創刊前・秋号をお届けしたら、いよいよ10月1日創刊です。いますぐ購読をお申込みください。

【購読料金】

●1年購読料金(12冊)………… 6,000円(1冊当たり500円)

●3年購読料金(36冊)……………12,000円(1冊当たり333円)

＊料金お支払いは、創刊号をお届けしてからで結構です。

発行：日経マグロウヒル社

〒101 東京都千代田区神田小川町1-1

読者サービス・センター ☎(03)380-3157

郵便はがき

168-□□

料金受取人払

杉並南局承認 55

差出有効期間 昭和60年2月28日まで (切手不要)

(受取人)
東京都杉並区浜田山4-5-5
杉並南郵便局私書箱35号
日経マグロウヒル社
読者サービス・センター
NBY係行

日経バイト NIKKEI BYTE 特典付(創刊前・秋号贈呈)・予約購読申込書

読者のプロフィールを正確に把握しニーズに応える雑誌づくりを進める資料といたします。どうぞ全項目にご記入ください。

お名前(フリガナ) 年齢 歳 自宅電話 市外局番 局 番 号

ご自宅住所(フリガナ) □□□-□□

お勤め先名 所属部署 会社電話 市外局番 局 番 号

ご購読期間(いずれかに☑を) □1年(12冊)6,000円(1冊当り500円) □3年(36冊)12,000円(1冊当り333円)

購読料金お支払方法(いずれかに○印を) 1.郵便局からの払い込み 2.銀行預金口座自動引落し

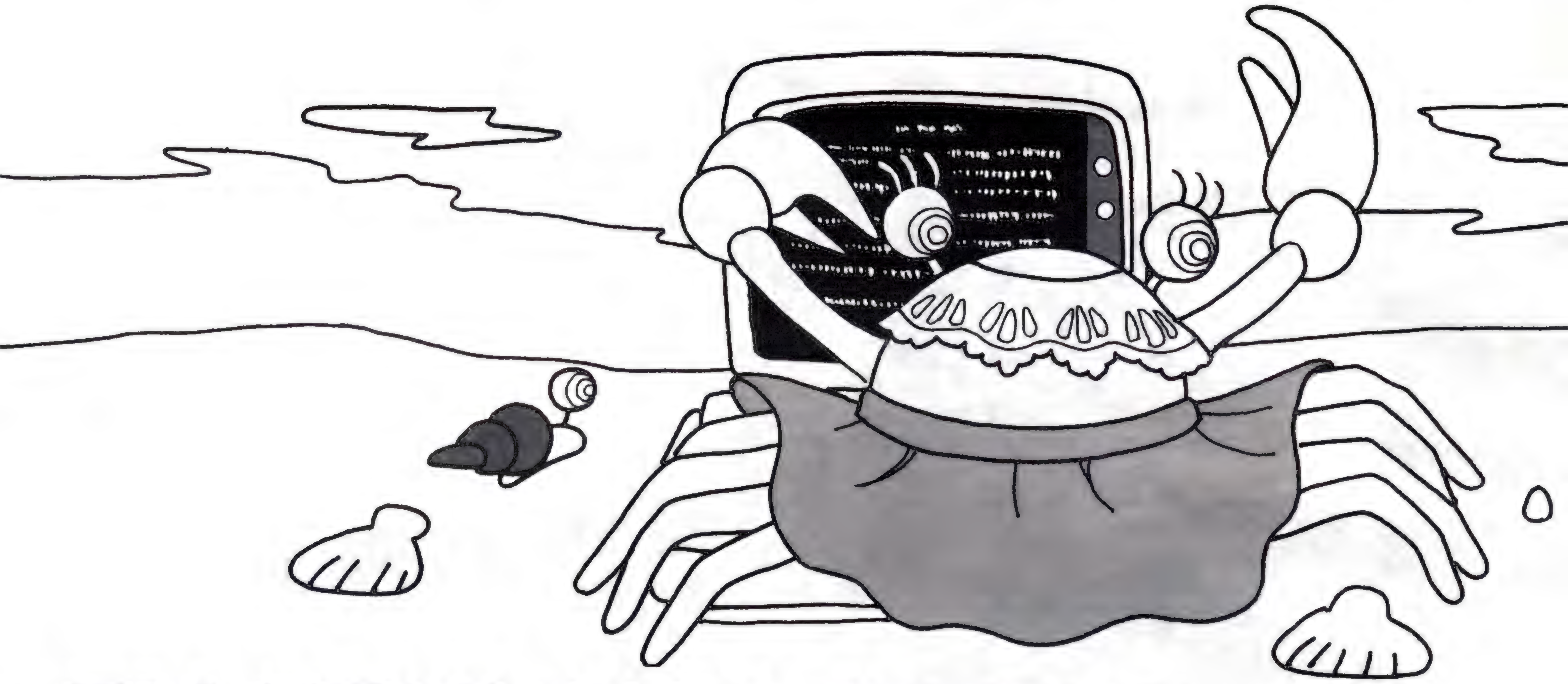
業種(具体的にご記入ください) 職種(該当のものに○印を) 1.コンピュータ(ハード)技術者 2.コンピュータ(ソフト)技術者 3.エレクトロニクス技術者 4.その他の技術者 5.プログラマー／アナリスト 6.専門職(弁護士,医師,会計士など) 7.教職 8.事務・営業 9.学生 10.経営・管理職 11.商工サービス自営 12.その他

Oh-FM 42-4633

書店ではお求めになれません。お申込みはこのハガキかお電話で。☎(03)380-3157

(日・祝日を含めて24時間申込専用電話)

カニでもカンタン。



FM-KOKYAKU

漢字顧客管理システム
FM-7.8 ¥30,000

32項目を自由にファイルフォーマット・変更もできる

■仕事内容

- ▶最大項目数：32項目(自由に設定・変更可能)
- ▶漢字・熟語：20,000語(人名・地名も含む)
- ▶最大登録件数：3840件(1件64バイトの場合)
- ▶検索：すべての項目に検索条件ができるほか、多重検索も可能です。
- ▶検索条件：比較区分(＝・＞＝・＜＝)全体区分(AND・OR)
- ▶印刷：一覧表(4種類)ダイレクトメール(4種類)
  印刷書式設定・変更が自由
  データの分類処理・順番設定可能
- ▶プリンタ：シリアルドットプリンタ・
  熱転写プリンタ

話題のヒットソフト好評発売中

日本語ワードプロセッサ　FM-7
FM-JWP/7　¥20,000
- ●対応プリンタ：シリアルドットプリンタ
- ●カナ・ローマ字入力、文節変換で20,000語の辞書装備
- ●1ドライブ及び2ドライブのどちらも使用できます。
- ●罫線は実線・細線・アンダーライン、文字は倍角・全角・半角
- ●豊富な編集機能で計算処理も可能です。

漢字版作表簡易言語　FM-7
FM-PLAIN/7　¥20,000
- ●集計表・一覧表・見積書などが誰でも簡単に作成できます。
- ●20,000語の辞書装備・1ドライブでも2ドライブでも可能。
- ●四則演算・分類・検索・複写など豊富な機能を装備。
- ●自動罫線入力機能により操作が簡単です。

日本語ワードプロセッサ　FM-7・8
FM-JWP/S　¥39,800
- ●対応プリンタ：シリアルドットプリンタ・ビジネスプリンタ
- ●1ドライブ及び2ドライブのどちらでも使用できます。
- ●20,000語辞書装備・熟語・短文登録可
- ●実線・細線・点線・アンダーライン、倍角・全角・半角

檀家管理システム　FM-7・8　基本部　基本部 ¥150,000
お寺さんのお手伝い　オプション ¥50,000
- ●1,000軒の住所・氏名等を出力でき、回忌の案内及び明細、宛名印刷ができます。
- ●お布施、寄付金、年会費等の実績管理が出来ます。
- ●過去台帳の登録、檀信徒の登録、弊社にて承ります。(有償)

［広］

タコならもっとカンタン。

FM-KOKYAKUは、項目桁数・データの型を自由に設定できますので顧客管理の他図書管理・その他登録・検索・印刷の作業にご使用いただけます。又、フォーマット済FD付ですのでお買上げいただいたその時からすぐお使いいただけます。

実用的な顧客管理です。

漢字の入力が簡単です。
日本語ワードプロセッサFM-JWP/7と同様、カナ／ローマ字漢字変換により、漢字の入力がとても簡単です。

実用性があります。
あらゆる業種に幅広くご使用いただけるように、データ入力画面、ファイルフォーマットが任意に設定できます。

印刷書式の設定が自由にできます。
3種類のタックシールに宛名書印刷ができるほか、一覧表は任意の項目を任意の順で自由に印刷することができます。

大量の情報処理が可能です。
一枚のフロッピーディスクに最大3,840件ものデータを登録でき、しかも高速でアクセスできます。

すべての項目に検索条件が設定できます。
すべての項目に検索条件が設定できるほか多重検索も可能です。

財務会計システム

PC-9801・9801F　MZシリーズ・FM-7・8・11用

PCA高級財務会計システム

- あなたの会社に合わせた科目登録や補助簿登録ができ、合計金額1,000億円未満の会計処理ができます。
- 主要帳票はオフコン並の専用用紙が用意されておりますので、銀行や税務署等にそのまま提出できます。
- 画面は漢字の大文字で表示され、非常に見やすくなっています。
- 日常良く使われる主要48科目は、ワンタッチ入力方式になっています。
- 入出金データーの入力は自動仕訳方式にする等、使いやすく、正確で、しかも主要部分は機械語処理により、トータル・パフォーマンスを追求しています。
- 処理範囲によりA．B．Cと3段階のシステムが用意され、必要なレベルを選択できます。
- 会計専門家数名のノウハウを結集したシステムです。

ピーシーエー㈱製漢字版：PC-9800シリーズ・FM-11
¥98,000 ¥148,000 ¥198,000

FM-7・8・11用

FM-ZAIMU

- 1枚のミニフロッピーディスクに最大10,000件もの仕訳が入力保存できます。さらにミニフロッピーディスクを複数することによって、最大14,000件の仕訳伝票処理に対応できます。
- 仕訳伝票入力時の摘要入力、勘定科目名の修正等、漢字の入力はカナ漢字変換方式が使用でき、漢字の取り扱いはとても簡単です。
- カナ漢字変換による摘要入力の他に固定摘要のコード入力や更新後の仕訳伝票の修正など随所に豊富な機能を取り入れ、高い操作性を実現しました。

三和ビジネス㈱製基本部**¥150,000**オプション**¥50,000**

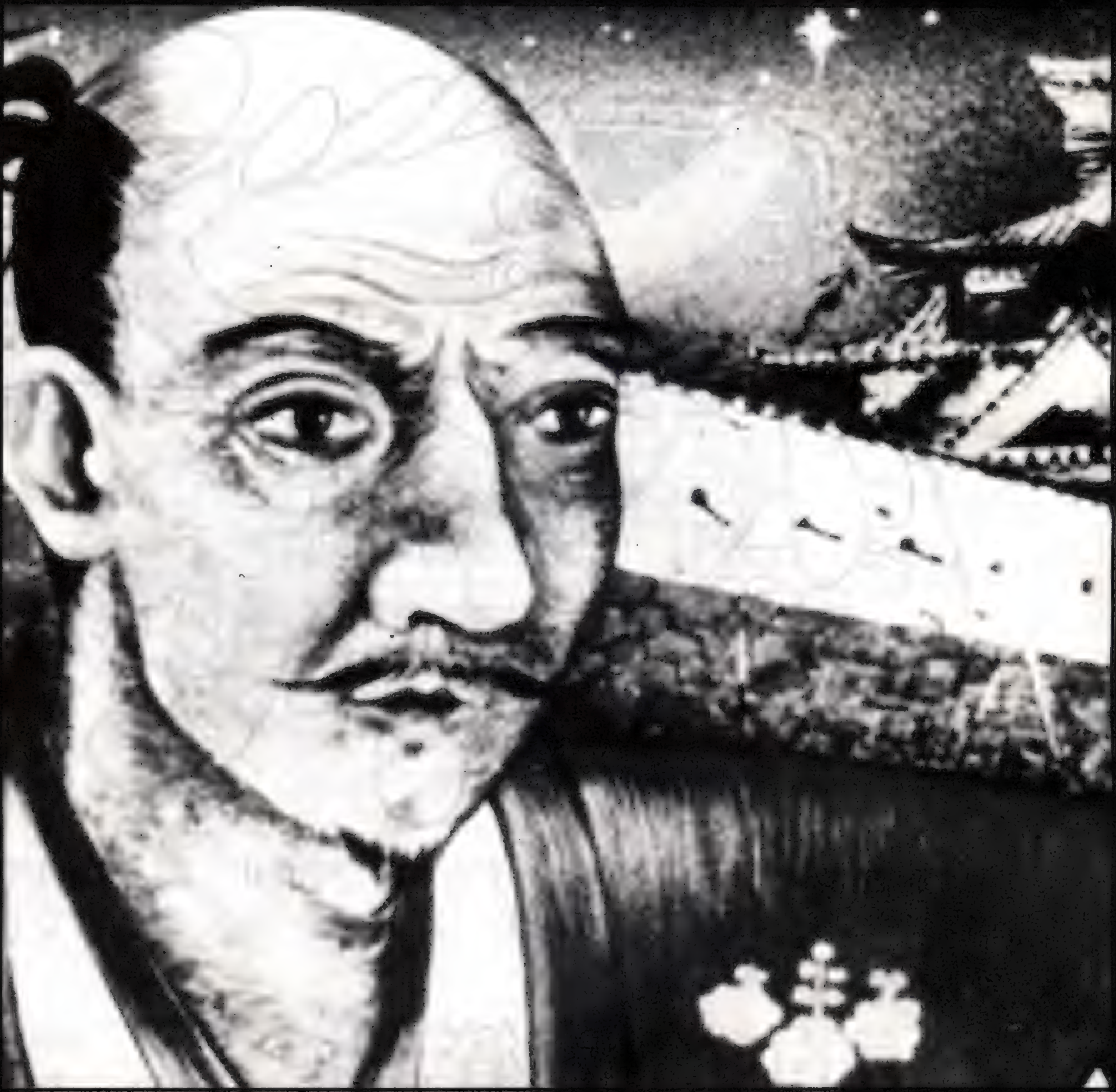
SOFTAWRE & BUSINESS・COMPUTER CO.,LTD.
SBC 株式会社 エス・ビー・シー
横浜市中区本町5-49（丸忠センタービル3F 〒231）
☎045-201-8480㈹　FAX 045-201-8386

●カタログ請求は上記まで

資料請求券 Oh! FM

KOEI

信長の野望



信長の野望

全国各地を猛将が割拠する戦国時代、彗星の如く出現し、天下統一という誰もが夢み果せなかった野望を胸に、戦国世に戦いを挑む時代の革命児、織田信長。
さあ今、貴方は「信長の野望」を実現できるか。
（もう目が血走っちゃいそう！信長につき合って二晩徹夜してしまったと言ったら読者は笑うだろうか。しかし、これホントの話。「ポプコム」より）

テープ版 4,500円 ＰＣ−8001/ｍｋⅡ、ＰＣ−8801/ｍｋⅡ
ＦＭ７・８、Ｘ１(Ｃ)、ＰＣ−6001/ｍｋⅡ、
ディスク版５″ 6,800円 ＰＣ−8001/ｍｋⅡ、ＰＣ−8801/ｍｋⅡ、ＰＣ−9801/Ｆ、ＰＣ−6601、
ＦＭ７・８・11(ＡＤ)（ＰＣ−9801用８″）7,800円
ディスク版3.5″ 6,800円 ＦＭ77

ホイホイ

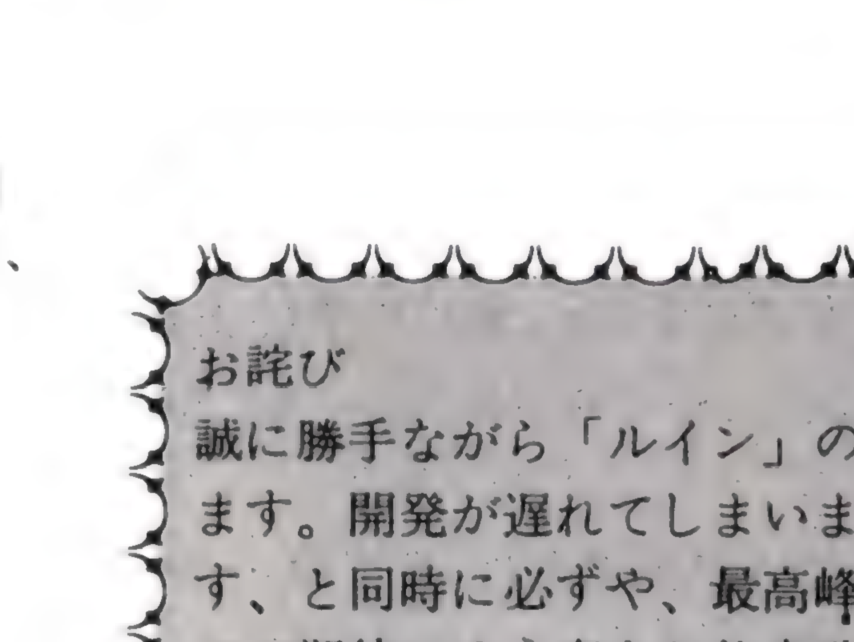
☆ＰＣ−6001ｍｋⅡ、ＰＣ−8001ｍｋⅡ用完成

爆笑度 100%のパロディ精神に脱帽！手法はあくまで正統的なウォーゲームを踏え、コミカルゲームに仕立てあげたこの手腕。文句なしのお買得の秀作といえよう！
「パソコンゲームランキングブック」より

テープ版、3,500円 ＰＣ−8801/ｍｋⅡ
ＦＭ7、Ｘ１(Ｃ)、ＰＣ−6001ｍｋⅡ、ＰＣ−8001ｍｋⅡ
ディスク版５″ 7,800円ＰＣ−9801

団地妻の誘惑

☆Ｘ１(Ｃ)、ＦＭ77用完成



ひょっとしたらこのゲームはノルマに追われる我々現代人への作者の強烈な文明批判ではないかという気さえしてくる。グラフィックや様々なアイディアで充分楽しめる上出来のソフト「カーソル」より

テープ版、4,800円 ＰＣ−8801/ｍｋⅡ、ＦＭ７、Ｘ１(Ｃ)
ディスク版８″ 7,800円 ＰＣ−9801
ディスク版５″ 6,800円 ＰＣ−8801/ｍｋⅡ、ＰＣ−9801/Ｆ、ＦＭ７・８・11(ＡＤ)
ディスク版3.5″ 6,800円 ＦＭ77

ナイトライフ

☆ＦＭ77用完成

テレビ、新聞、雑誌にとこれ程話題に上ったソフトは、かってなかった、世界に類を見ないアダルトの世界
御結婚のお祝いにもぜひどうぞ！

テープ版、4,500円 ＰＣ−6001ｍｋⅡ、ＰＣ−8001、ＰＣ−8001ｍｋⅡ、ＰＣ−8801/ｍｋⅡ、ＦＭ7・8、Ｘ１(Ｃ)、ＭＺ2200
ディスク版８″、8,800円 ＰＣ−9801
ディスク版５″、7,800円 ＰＣ−6001ｍｋⅡ
ＰＣ−8001、ＰＣ−8001ｍｋⅡ、ＰＣ−8801/ｍｋⅡ、ＰＣ−9801、ＰＣ−9801Ｆ
ＦＭ7・8・11(ＡＤ)
ディスク版3.5″ 7,800円 ＰＣ−6601、ＦＭ77

お詫び
誠に勝手ながら「ルイン」の発売を10月中とさせていただまます。開発が遅れてしまいました事を深くお詫び申し上げます、と同時に必ずや、最高峰のゲームとなって発場し、皆様のご期待にそう事をお約束致します。

代表取締役社長　襟川　陽一

KOEI

☆テープ版を10倍バージョンアップしてディスク版堂々完成!!

Dungeon

ダンジョン

今、アメリカで最も人気があり、ゲームの主流であるロールプレイングゲーム。これを思考ゲームの殿堂、光栄が創るとどうなるか。もちろんテープ版の極限もきわめてありますぞ。さあ、ダンジョン(地下迷宮)へ旅立つ貴方、光栄のファンタジーロールプレイングゲームでこのだいごみを十二分にお楽しみあれ!(ゲーム進行も比較的シンプルだし、ストーリーも日本人にとって理解しやすい。ロールプレイングゲーム入門に最適。)ログインより

テープ版 5,200円(プレイヤーズテープ付)ＰＣ-8001/mkⅡ、ＰＣ-8801/mkⅡ、ＦＭ７
ディスク版5″ 7,800円 ＰＣ-8801/mkⅡ、ＦＭ７
ディスク版3.5″ 7,800円 ＦＭ77

☆大絶替のペナントレースＰＣ-9801(Ｆ)
ＰＣ6001mkⅡ、ＦＭ77遂に完成!!

ペナントレース

ペナントレース

このソフトはスゴイ。ちゃんと守備では前進、定位置の切り換えができるし、バントシフトもとれる、これは秘密だけど、すごい魔球も。このペナントレースをやらずして野球ゲームは語れない!
「テクノポリス」より

テープ版、 4,500円 ＰＣ-6001 mkⅡ、ＰＣ-8001/mkⅡ、ＰＣ-8801/mkⅡ、ＦＭ７、Ｘ１(Ｃ)
ディスク版５″ 6,800円 ＰＣ-6001 mkⅡ、ＰＣ-8001/mkⅡ、ＰＣ-8801/mkⅡ、ＰＣ-9801Ｆ、ＰＣ-9801、ＦＭ７・11(ＡＤ)
ディスク版3.5″ 6,800円 ＰＣ-6601、ＦＭ77

商売繁盛

ＦＭ７・８用

汎用 漢字顧客管理システム￥29,800

- 1ディスケットあたり最大600件の顧客情報(名前、住所、電話番号、年令、性別等)が仮名漢字変換方式により、簡単に漢字で登録できます。
- 業種に合わせたユーザー独自の項目設定可能。
- メニュー方式による容易な操作。
- 4つまでの多重検索条件の設定が可能。
- 宛名書き機能により漢字でダイレクトメールが発行できます。
- 豊富なプリンタサポート。
- 現在市販されている殆んどのドットプリンターで漢字の宛名書きが出来来ます。

株式会社 光 栄
本社 〒326 足利市通１丁目２６７７ ☎0284-41-5911(代)
支社 〒223 横浜市港北区日吉本町1876 ☎044-61-6861(代)

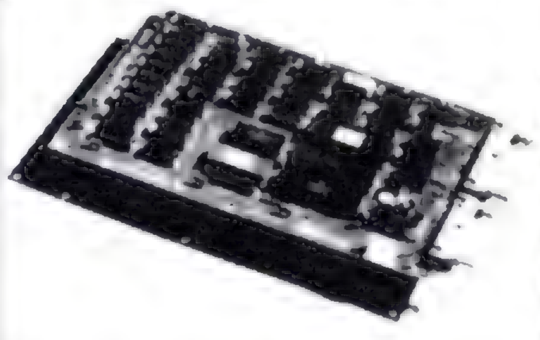
☆お求めは全国のマイコンショップ又は本社宛現金書留にてお送り下さい。(送料サービス)

カタログ請求

68系のSoft＆
Hardを開拓する

macro systems

MACRO-9 TypeII

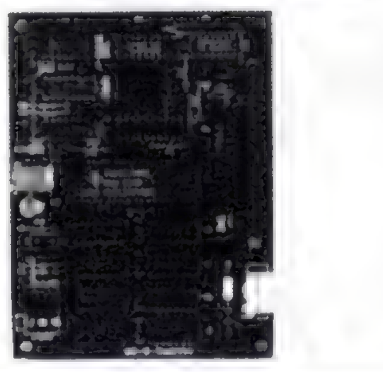


■FLEXが極限まで使いこなせる高速シングルボードコンピュータ(トラ技別冊"6809オールマイティ参照")

☆6809のソフトウェア開発に最適。

- ●主要チップ　CPU：68B09、ACIA：68B50、FDC：1793－02、VIA：6522A
- ●RAM：64Kバイト、ROM：8Kバイト　●5/8インチ、片/両面、単/倍密すべて対応.
- ●プリンタI/F：セントロニクス準拠　●シリアルI/F：RS-232C、300～38.4Kボー
- ●電源5V/1A　●ボードサイズ：145×249mm
- ●専用EPROMライター(2716～128用)14コマンド・ソフト付……………… ¥49,800
- ●MONITOR-M/Mコンパイラ付完成ボード……………… ¥149,800
- ●モニターROM、DC/DC CONV付ボードのみ……………… ¥14,800

MACRO-8　MACRO-8-28 VDP BOARD

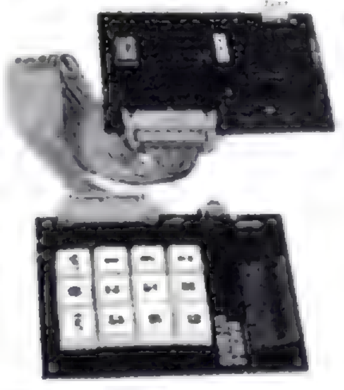


完成ボード
¥39,800

■VDP＋64KDRAMの組み合わせは高速アニメを実現します。

- ●256×192ドット、16色表示、4画面(VRAM＝64Kバイト)
- ●コンポジットビデオ出力、外部ビデオ入力あり.
- ●32のスプライト(動画面)を持ち、簡単なソフトで3次元的な効果が得られます
- ●スプライト中のキャラクタは座標の設定だけで自由に移動、拡大ができます.
- ●I/Oバスは26P(MACRO-9)、50P(MACRO-8、汎用)でヘッダーコネクタで接続します.
- ●ASCII/JISのキーボード入力があります.
- ●電源5V/500mA　●ボードサイズ：100×130mm

MACRO-8　MACRO-8 EV1



¥35,800
(キット)

■6809CPUボード＋KEYボード、キット

☆6809のマシン語、アーキテクチャの学習に最適。

- ●表示：7桁、7segLED(データ、アドレス、コマンド表示)
- ●KEY：12個(SIFTモードあり、23KEY相当)
- ●コマンド：10個(メモリチェンジ、イニシャライズメモリ、ムーブ、トレース、テレコのセーブ、ロード、ベリファイetc)
- ●カセットI/F(600～4800ボー)　●メモリバックアップパーツ別売
- ●MS-6809BUS(拡張用50Pバス)があります
- ●電源5V/500mA(CPU＋KEYボード)

MACRO-8　MACRO-8 EV2/EV2R

■6809CPUボード＋SERIALボードキット

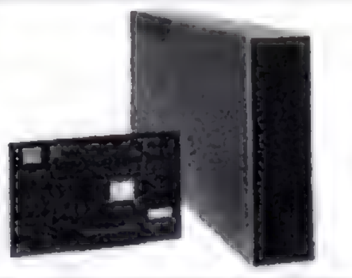
☆すでにパソコン(PC-8001MKII、FM-8、レベル3)、シリアルターミナルをお持ちの方へ。

- ●モニター：MONITOR-M
- ●基本コマンド：14、サブルーチン：35(SWI2コール)
- ●ボーレット：110～9600ボー
- ●MS-6809BUS(拡張用50Pバス)があります。●電源5V/500mA
- ●EV2(TTLレベルシリアル)……………… ¥31,800(キット)
- ●EV2R(RS-232C仕様)……………… ¥35,800(キット)

★FM-7用、各種I/Fカードはすべて高速モード(2MHz)で動作します。(日立SI用アダプタ近日発売します。)

FM-7　FM-7用シングルフロッピーディスクセット

FD-7 (MACRO-7-5付) ¥99,900



□FM7, MACRO-9TYPE2用RAMディスク
●FD0(MACRO-7-10別売)………… 予価¥99,800

□拡張用ディスク
●MACRO-FD5(増設用、ミニディスク)…… ¥74,800
●MACRO-FD8(増設用、標準ディスク)… ¥149,000

□拡張用ソフトウェア(ディスクドライバー＋ユーティリティ)
●E-FLEX：「FLEX-09用、片面・両面、単密・倍密を自動選択でき、また標準ディスク2台も同時に使用可能です(倍トラックも可)」
●E-CP/M：　各¥12,000

FM-7　ユニバーサル・ボード(MACRO-7-UV)……………… ¥2,800

32PI/Oスロット用のユニバーサル・ボードで、アナログ回路にも使用可能な様にスルーホールなしの両面基板です。

FM-7　フロッピー・ディスク・I/F(MACRO-7-5)……………… ¥39,800

3・3、5・8インチ、片面・両面、単密・倍密、及び高密度タイプのディスクも使用可能なVFO回路内蔵のフロッピー・ディスク・I/Fカードです。(5インチ両面倍密モードは、純正品と互換性があります)

FM-7　ROM・RAMカード(MACRO-7-6)メモリなし……………… ¥9,800
Mコンパイラ(アセンブラ含む)エディタ・DOS(MFLEX)・ROM付……… ¥69,800

2732・2764・27128タイプのROMまたは、6116・6264タイプのRAMのいずれかのメモリを最大4個(64Kバイト)まで実装できるROM・RAMカードです。また、メモリ・ロード用の32バイトのプログラムも実装(禁止も可)されており、最大3枚まで使用可能です。(RAMバック・アップ用のパターンあり、パーツなし)また、RAM実装時にF・BASICのソースを保存する為のプログラムリストが付いています。

FM-7　BOOT・RTCカード(MACRO-7-7)……………… ¥39,800
(RTC部、ROM・RAMなし、パワー・オン回路のみ付)……………… ¥19,800

電源オンまたは、リセット時にユーザープログラム(MACRO-7-6に実装されているROM内のプログラムなどを実行できます。RTCなどが実装されている物は、起動時にFM-7本体内の時計に自動的に時間を設定できます。また、$FE00～$FFDFまでのIPL, ディスク・ドライバー・プログラムがRAM(4バンク)にもプログラムできますから、RAMディスク(MACRO-7-10など)・ハードディスクの(MACRO-7-11)使用が容易になります。　RAM及びRTC部は、バッテリィ・バックアップを行なっています。(40Pのスロットに実装します。)

FM-7　高速シリアルアダプタ(MACRO-7-10)……………… ¥19,800

別売のRAMディスク(MACRO-FD0)と組み合わせると320Kバイト(5.25インチ両面倍密1枚分)の高速サイレント・ディスクになります。また、75bar～76.8KbarまでのRS-232Cの入出力が可能なパターン(パーツなし)もあります。最大4枚まで実装できます。(68B50　2個付)

FM-7　A/Dコンバータ・カード(MACRO-7-12)……………… ¥19,800

8ビット90μs(10ビット100μs)、16チャンネル、0～5VのA/D変換が可能で、13μsで応答するプログラム電圧比較機構(8ビット)もあります。また、未使用なPIAのパターン(パーツなし)もあります。最大8枚まで実装できます。

FM-7　EPROMライタ(MACRO-7-8)……………… ¥19,800

●Mコンパイラのwオプションで直接呼出し可能なプログラムCP/M、FLEX、F・BASIC用… 各¥5,000

2764、27128、27256の高速書き込み(Vcc6V)及びVppの12V仕様にも対応できるEPROMライタ・カードです。2716、2732は、リードのみ可能で、最大8枚まで実装できます。(プログラムリスト付)

FM-7　PIA・ユニバーサルボード(MACRO-7-9)PIAなし……………… ¥9,800
68B21 PIA 2個付……………… ¥12,800

入出力方向の設定が可能な40 I/Oポートの他に28mm×114mm(12×46穴)のユニバーサル・エリアを持つボードで、最大8枚まで実装できます。

FM-7　ハードディスクアダプタ(MACRO-7-11)……………… 予価¥39,800

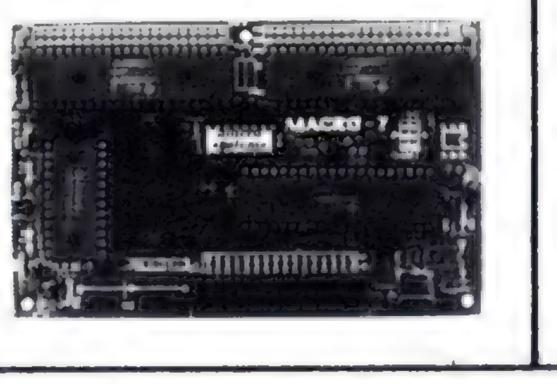
別売のHD-5と組み合わせることで、FM-7でハードディスクの使用が可能になります。

68B09　MACRO-7CPUカード(68B09、68B21×2、2KRAM付)……………… ¥19,800

6809、6821×2、6264(または6116)、27128(または2764)が実装可能で、電池などの追加でRAMのバックアップも可能なCPUボードです。32Pのバスを持っていますのでFM用の各種IFカードが使用できます。

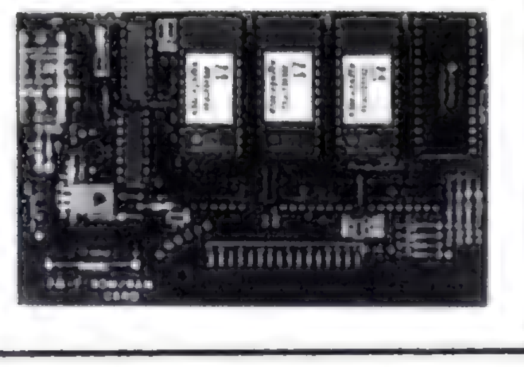
MACRO-7　CPUカード

☆ソフトが完成したらあとはおまかせ究極のボードコンピュータ! (ソフト開発はMコンパイラが最適です。)



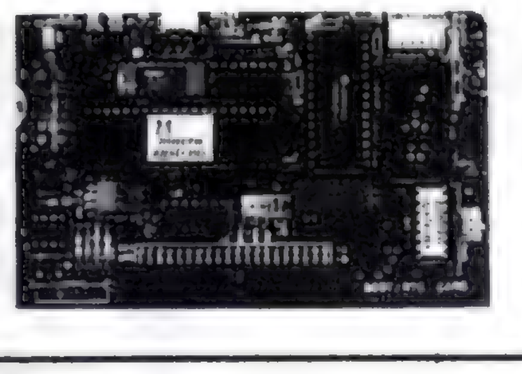
MACRO-7-6　ROM/RAMカード

☆あなたの好きなソフトを乗せてください。ROMでもRAMでも結構です。CMOSメモリとバッテリを使えばバックアップもできます。(オプション)



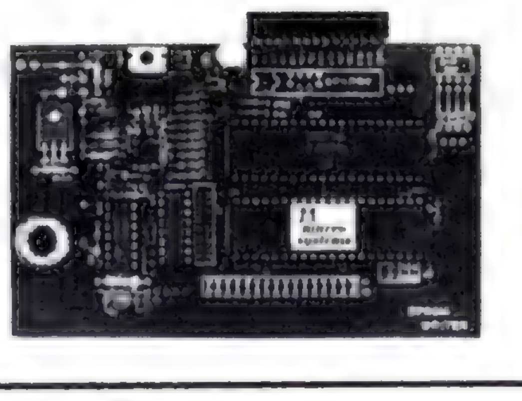
MACRO-7-7　BOOT/RTCカード

☆あなたのFM-7を確実に進化させる、タイマー、リアルタイムクロック、パワーONローダー等々!



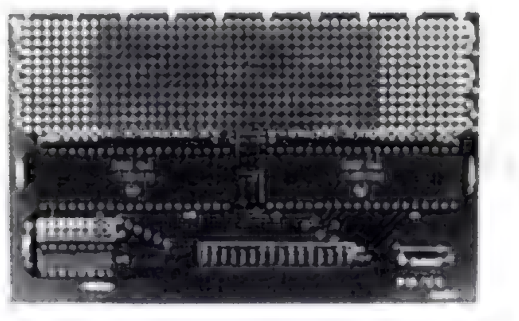
MACRO-7-8　EPROMライター

☆2764Aが15秒程度できれいに焼けます。超高速EPROMライター!



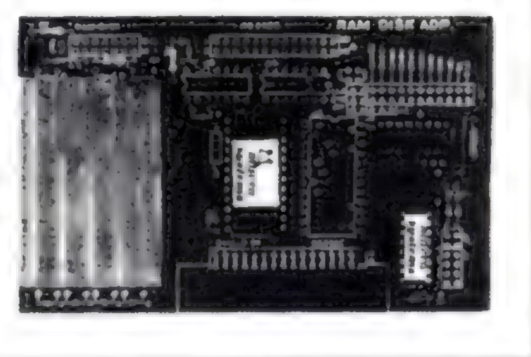
MACRO-7-9　PIA/UVカード

☆PIA(68B21)が2個とフリーエリアを持ちます。自作派からシステム制御まで!



MACRO-7-10　高速シリアルアダプタ

☆本当はRAMディスクアダプタなんですが、ACIAのソフトを作ると高速シリアルI/Fに早替りします!



もし、あなたがあなただけのシステムを必要としているなら、マクロシステムズへお電話ください。基板アートワークからシステム開発、試作から量産までお引受けいたします。他社とは一味違う製品をおとどけできるでしょう。もちろんCPUは、6809で!

貴社のニーズに合わせた6809のソフト、ハードを開発いたします。

マクロシステムズ株式会社

住所：〒101東京都千代田区外神田3-4-1(外神田ビル)☎(03)251-5148
振 込 先：三菱銀行秋葉原支店〈口座番号〉4525872(普通)9022818(当座)

資料請求は左記住所まで御連絡ください。

取扱店
〈秋葉原地区取扱店〉本多通商(株)☎251-7611・253-6465　(株)若松通商マイコンセンター若松☎251-8933　マイコンパーツセンター☎253-8521　〈大阪〉共立電子産業(株)シリコンハウス共立☎(06)644-4446　〈名古屋〉中部本多通商(株)☎(052)263-1670　〈九州〉九州本多通商(株)☎(092)713-8018

※MACRO-9TYPEIIは、ほとんどのFLEX-09をブートできます。
※MACRO-7-5によって使用できる5.25インチ両面倍密度ディスクは、純正品と同じようにヘッド・ロード機能が必要です。
※MACRO-7用マザーボード(MB-SI、FM-7対応)開発中。
※各仕様は改善の為、予告なく変更されることがあります。
※電話でのお問い合せは、午後1時までにお願いします。

Thinking Recreation

STON BALL

ストン・ボール

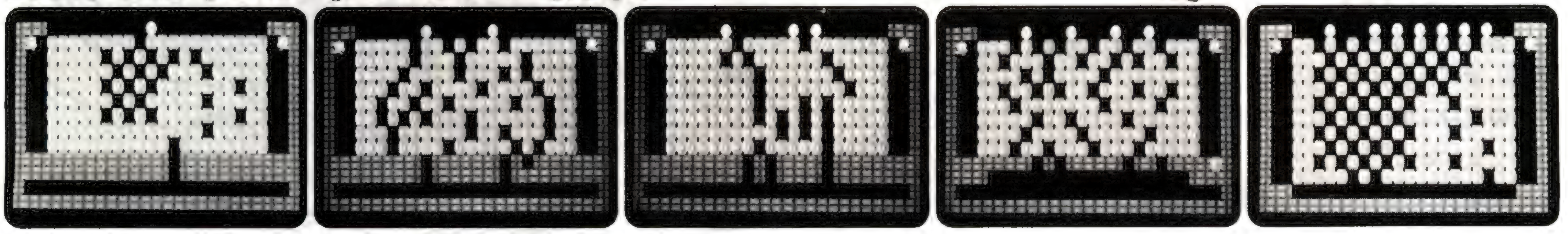
(〒240円)
定価3,800円

左右2人のかわいいキャディ坊やを上に下に操りながらブロックをひと押し、ふた押し。ストン、ストンと音をたててブルーのボールが落ちていく。底まで落とせば大成功、さらに難解なパターンが現われる。
複雑怪奇なブロックの組み合わせ、2つ3つ…と増え続けるボールがキミを苦しめ、20面には何とボールが10個。これを底まで落とすのは至難の業。これをキミは何時間、いや何日(?)で解けるか。1度始めたらディスプレイに釘付け。時の経つのも忘れて熱中してしまうユニークなパソコン・パズルにチャレンジ!

FM-7 PC-8801/mkII PC-9801 E/F X1/C

キミは何面クリアできるか
超難解
20面の10ボール・ドロップに挑戦!

難解



実践ゲーム制作テクニック・ノウハウ集

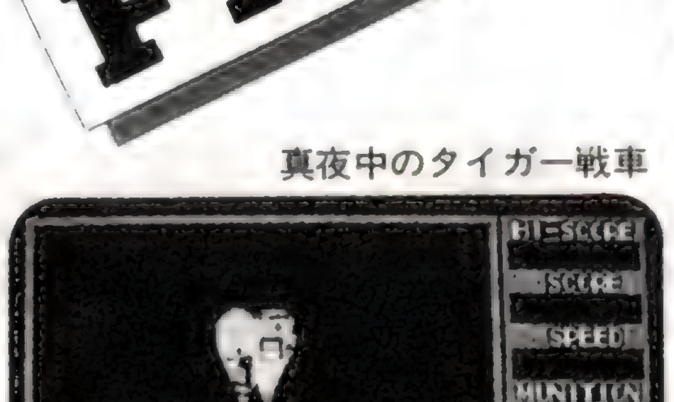
FM-7

MIAテクニカル・ライブラリ[1] [書籍&カセットテープ]

マシン語 高速ゲームの制作

A5判
定価3500円(〒350円)

真夜中のタイガー戦車



「FM-7で面白い高速リアルタイム・ゲームを作りたい」という人のために、6809アセンブリ言語の基本、サブシステムの使い方、特殊な画面処理(仮想VRAM)・重ね合わせの処理等、プログラミングに役立つ実践テクニックをご紹介。
(FM-7、リアルタイム・ゲーム付)

- ●STEP1/アセンブリ言語について…6809アセンブリ言語の特徴
- ●STEP2/サブシステムの使い方…共有エリア/メインテナンス・コマンド/コマンド・ハンドラ/サブシステムへのプログラム転送
- ●STEP3/ゲーム作りに有用なサブルーチン群…カラーパレットの設定/PSGコントロール
- ●STEP4/画面処理…グラフィック・パターンの表示、移動、重ね合わせ/仮想VRAM
- ●STEP5/ゲーム・プログラム

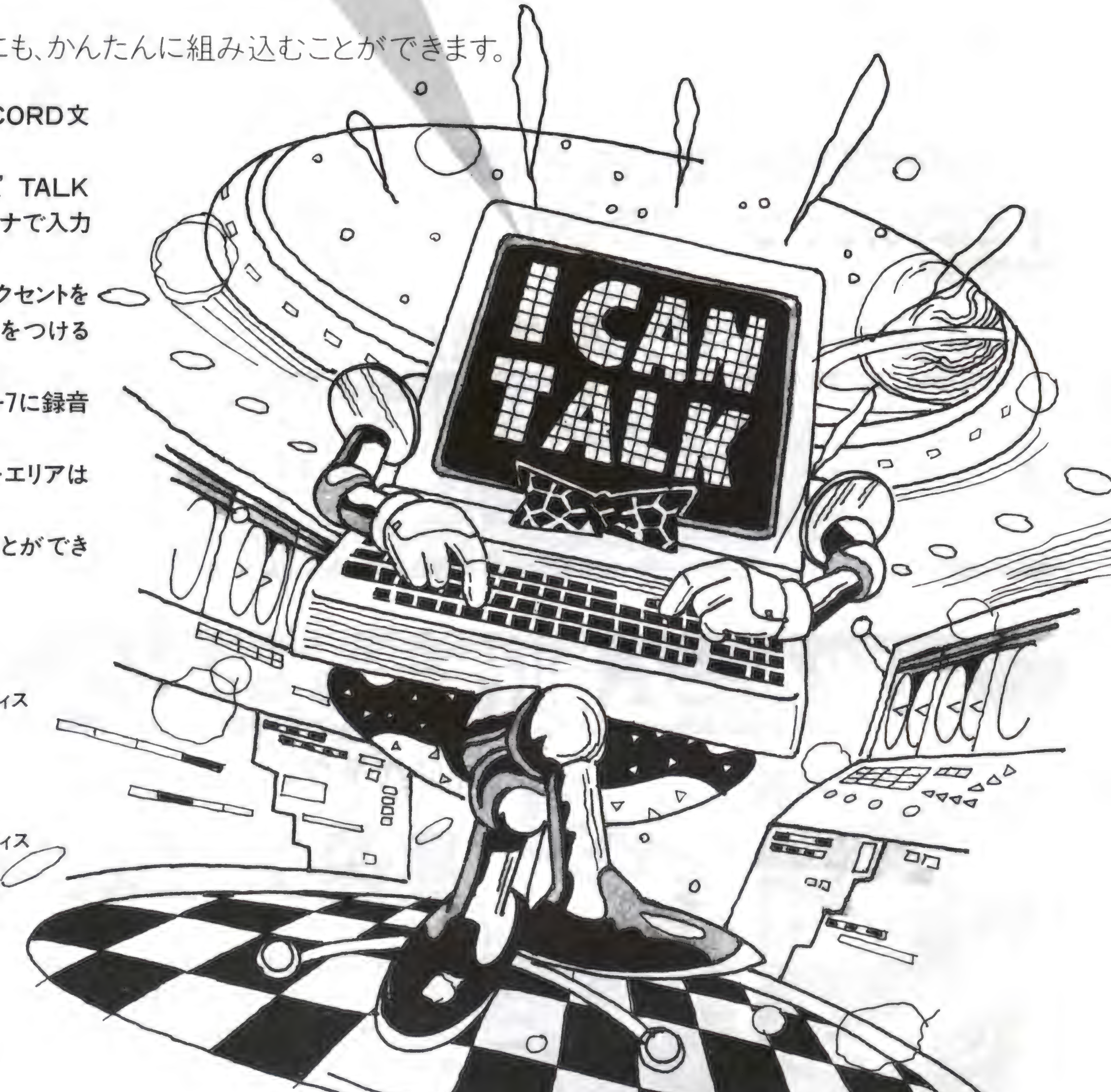
■お求めは最寄りのマイコン・ショップ、書店へ。または郵送料を添えて下記へお申し込みください。
〒102 東京都千代田区紀尾井町3-20 TEL.(03)265-2461 ㈱エム・アイ・エー

MIA
MICRO INFORMATION ASSOCIATES

やれば、できる。

FM-7のおしゃべり機能
HuVOICE™ 新発売

FM-7ご愛用の皆さんへニュース。
なくてくやしい、と思っていた機能の新発売であります。
HuVOICE——ついにFM-7が話すことを覚えます。
ハードの改造は一切不要、
しかもBASIC。
いままでつくったプログラムにも、かんたんに組み込むことができます。



- ●BASICを拡張、TALK文・RECORD文が使用できます。
- ●TALK文は、TALK“ハドソン”TALK“エフエムセブン”のように、カタカナで入力するだけでOK。
- ●数字を指定するだけで言葉にアクセントをつけることができます。また音階をつけることもできます。
- ●あなたの声をマイクを通して、FM-7に録音することができます。
- ●裏RAMを使用しますので、フリーエリアはまったく損ないません。
- ●基本システムのみで使用することができます。

HuVOICE™〈カセットバージョン〉
¥6,800(2本組)
最低必要なシステム——FM-7本体/ディスプレイモニタ/データレコーダ

HuVOICE™〈ディスクバージョン〉
¥9,800
最低必要なシステム——FM-7本体/ディスプレイモニタ/ディスクドライブ/F-BASIC V3.0

HUDSON GROUP
HUDSON SOFT®

※HuVOICEは、FM-7専用です。また音声をFM-7に録音する場合には、マイクとアンフ、またはテープレコーダーが必要となります

出るか!! 日本アドベンチャー・ゲームの極めつけ

すでに米・東海岸のディズニー・ワールドの取材を完了

東京ディズニーランドには何回か足を運んで、本場のディズニーも見物しちゃった！なんて方は結構多いんだネ。でも、南部フロリダの“ディズニー・ワールド”は意外と知られていない。まあッ、そのドデカさたるや、東京都がスッポリ入っちゃうってくらいのスケール。やっぱりアメリカならではのことなんだ。

そのディズニー・ワールドに目を付けたのが、ハドソンの天才男・中本伸一と竹部隆司。どこで聞きつけたのか、「ウーム、これだけのスケールと面白さなら、なんか面白ゲームの素材になるんじゃないの。いっぺん、取材にいくか」なんぞと、意気投合。ついに4月末成田からの出発となったんであります。

西海岸へ寄って、さらにニューヨーク、そこから5時間ほどかけての現地入りでした。あらかたの資料は読んできたものの、その広大さに2人して顔を見合わせ、「おい、俺たちいったいどこから手をつければいいんだろうネ」っていったかどうか、まあ、そのへんの細かいことはこの際置いとくことにして、2人はただただ驚嘆するばかり。

とにかく、この一大リゾートにはフリーウェイが突走り、モノレール・システムが縦横に動き、来場者は15分おきに出るバスを利用するってんだから想像がすこしはつくだろう。内容は大きく分けてエプコット・センター、レーク・ブェナ・ビスタ・コミュニティ、それにバケーション王国の3大地区で構成されている。

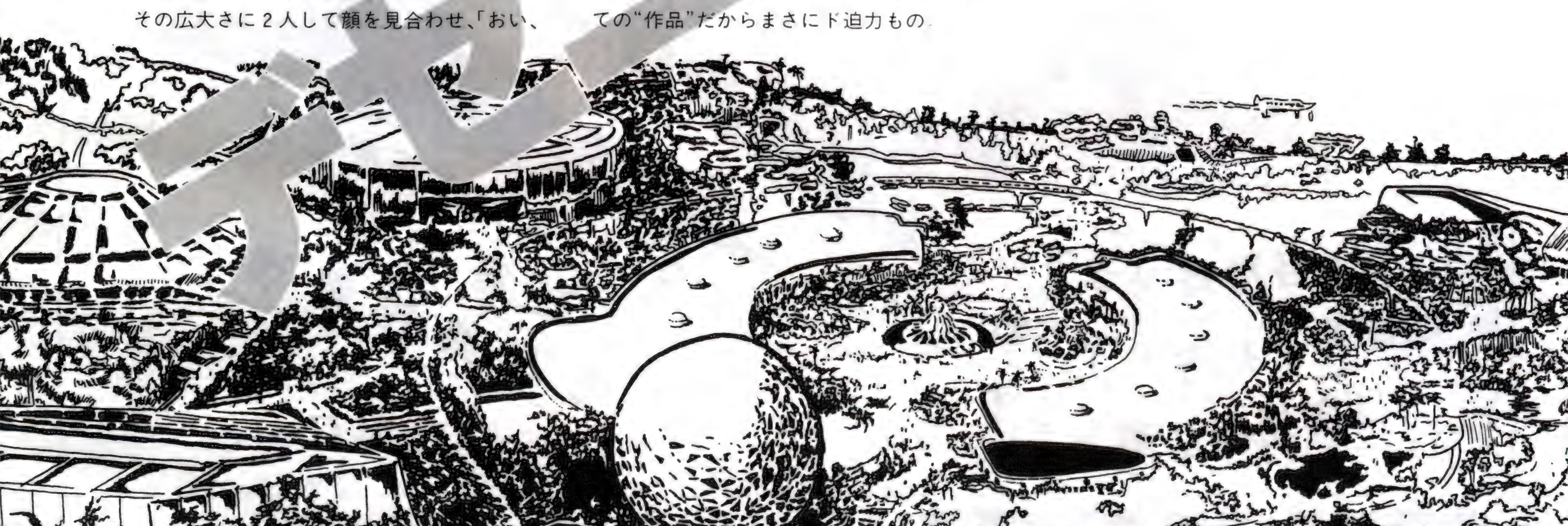
なかでも、中本・竹部の2人が注目したのがエプコット・センター。ここには、通信、エネルギー、交通などといった各システムが、見事に未来社会を構築している。それも、ちゃちな遊園地みたいに単なるお飾りじゃなくって、アメリカの大企業が、持てる最先端の技術をおしげもなく投入しての“作品”だからまさにド迫力もの。

自然と親しむものだって、もり沢山。中本氏、「ディスカバリー・アイランドではあんまり暑いものだからつい泳いじゃった」というくらい、ここには素晴らしい人造湖もある。

そんなわけで、仕事と称し取材に行ったのか遊びに行ったのか、なんだかわからないのだけれど、“天才男”のことだから、なにかモノになるのだろう。

いまひそかに計画中の案は、このディズニー・ワールドを日本に作っちゃおうという話だ。それも、あのタモリが云々いう、“ナーゴヤ”に。机上の一大リゾートであるけれども、彼ら2人の頭の中からは、フロリダのディズニー・ワールドのイメージがなかなか消えない。どんな作品が飛び出してくるか、すでにデゼニワールドの作品名で制作に入っている。

デゼニワールド



HUDSON GROUP
HUDSON SOFT®

ハドソン札幌/〒062 札幌市豊平区平岸3条5丁目4番17号 コロナード平岸II 201 PHONE:011-821-1538
ハドソン仙台/〒980 宮城県仙台市宮町1丁目4番28号 PHONE:0222-65-7031
ハドソン金沢/〒920 石川県金沢市本町2丁目1番28号 PHONE:0762-23-1263
ハドソン東京/〒102 東京都千代田区麹町4丁目7番5号 麹町ロイヤルビル2F PHONE:03-234-4996
ハドソン大阪/〒542 大阪市南区南船場4丁目2番18号 佐野屋橋ビル5F PHONE:06-251-1945
ハドソン福岡/〒816 福岡県福岡市博多区博多駅東2丁目4番301号 いわきビル103号 PHONE:092(441)4568
ハドソン沖縄/〒901-21 沖縄県浦添市大平8番10号 PHONE:0988-79-3361

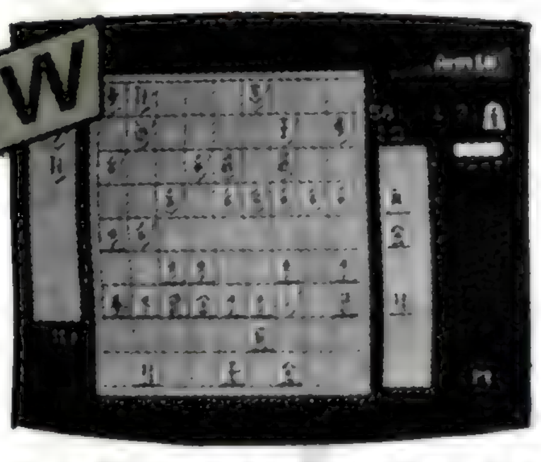
LPガス販売管理
及び
ボンベ管理システム
特長その1
入力画面及び印刷帳表類が
全て漢字で扱われている
ため内容の把握が
一目瞭然。
特長その2
数字の顧客コードを顧客
名の読みと同様に使え
るのでコードを覚え
る必要がない。
コンクリート技術計算
システム
他各種試験
X管理図
ヒストグラム
P管理図
品質管理
販売管理システム
漢字処理がオフコン並！
日報出力からダイレクトメール
の宛名印刷、請求書発行まで
ボタン一つで自由自在
オーダーメイドソフト
各種承ります。
仕様書をお送り下され
ば御見積り致します。
ガソリンスタンド販売管理
在庫管理
宅配業務管理
e.t.c…
FUJITSU
FMシリーズ専門ソフトウエアハウス
株式会社 システムセンター三重
〒510 三重県四日市市日永西三丁目4−1
TEL 0593−47−2244

FM-7は魔法のランプ!

ソハコムはコンピュータイレブンの
オリジナルソフトウェア・ブランドです

★本格派将棋ソフト「名将」
対局&詰め将棋で楽しさ2倍

NEW



史上最強の対局将棋ソフト、ついに登場。通信機能(Disk版)をひっさげて君にチャレンジ。しかも、自分で問題入力ができる詰将棋ソフトまでついてきてこの価格。今までの将棋ソフトにあきたらない君、さっそく名将にチャレンジしよう!

FM-7、FM-77、FM-NEW7用

- 5インチDisk版　5,800円
- 3.5インチDisk版　5,800円
- カセット版　3,800円

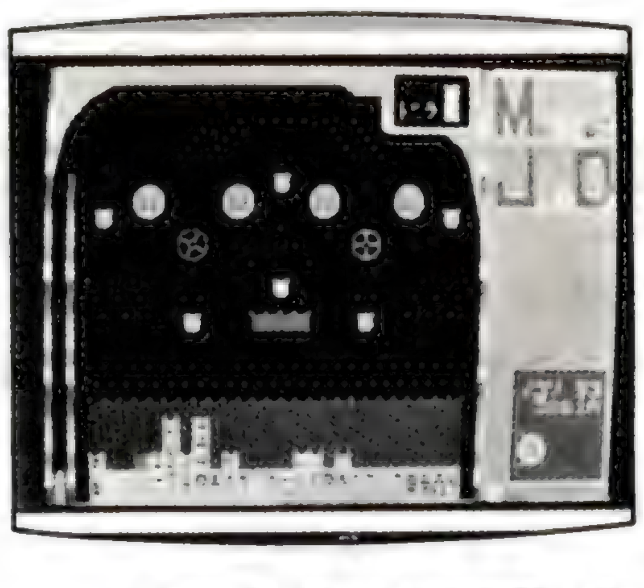
★ 重機動戦隊 シグナス
——傑作アクション・ゲーム——

●君は地球防衛隊、総指揮官は地球を守れ!

5インチ・ディスク版
¥6,800

(PC8801mkⅡ、FM-7、FM-NEW7対応)
※FM-77、3.5インチ版／近日発売予定

★マスカット Jan-Q
あの雀球がJan-Qになった!



オールマシン語だから超高速! 玉の動きも部品の動きも雀球そのまま! あなたはパチンコ屋で味わう以上の面白さ、楽しさを感じるだろう! FM-7の壁を打ち破る画期的ソフト遂に完成!

FM-7、FM-New7用(テープ版)　定価3,800円

★実戦MSXアセンブラ入門
プログラムをつくりながら覚えよう!

NEW

MSXによるマシン語入門決定版。Z-80の基礎知識から、ビデオプロセッサ、サウンドジェネレータの扱い方まで、実際にプログラムで確かめながら覚えられる。アセンブラ、遊んで覚えりゃ恐くない!

B5判／182ページ
定価1,800円(〒300円)

★FM-7 GAMEで覚えるBASIC RUN
プログラミングが好きになる!

難しいことは一切抜きの画期的なBASICプログラミング入門書。イラスト、チャートをぜいたくに使用。楽しいゲームをしながらBASICをマスターしちゃうなんてズルイ、ズルイ。(1,000円にてカセットサービス有り)

B5判/140ページ
定価1,800円(〒300円)

★誰にでもわかるFM-7 6809マシン語入門 改訂版
6809マシン語はこれでバッチリ!

6809マシン語の知識がまったくない人を対象にした名著の改訂版です。まったくわからない人にもわかるから、ちょっとわかる人はもっとわかるはず。

B5判／170ページ
定価2,800円(〒300円)

★アドベンチャーゲーム・ガイドブック
(仮称)

近日刊行

今、話題のアドベンチャーゲームのヒントが詰めこまれている。迷路へはまった君への貴重な資料だ。満載ゲームは「トランシルバニア」「シャーウッド・フォレスト」「ユリシーズ」「鍵穴殺人事件」など全11ゲーム。

★その他の書籍

- マシン語GAMEのつくり方(FM-7/8、PC-8001、L-Ⅲ用)……¥2,800
- FM-7/8マシン語の本……¥2,800
- 6809アセンブラ……¥3,800
- 6502アセンブラ……¥4,000
- パーソナルコンピュータのデータ処理……¥2,800
- おもしろまじめF-BASIC……¥2,800

キミの作ったプログラムやキミの書いた本を商品化します。お気軽にお問い合せください。

——お問い合せは…㈱日本ソフト&ハード社出版局まで——
〒160 東京都新宿区高田馬場2-8-3 生沼ビル2F　☎03(209)2585㈹

書店様、パソコン ショップ様へ——

このページに掲載されている商品は、小社商品営業課でもお取り扱いしております。

■お問い合せは…㈱日本ソフト&ハード社 商品営業課 ☎03(03)0612㈹
〒160 東京都新宿区高田馬場1-26-5 FIビル7F

コンピュータ・イレブンの
㈱日本ソフト&ハード社

出版部／03(209)2585 高田馬場本店／03(209)7376 新宿店／03(342)4821 横浜店／045(312)4611
名古屋店／052(451)7371 大阪ニュー梅田店／06(316)1171 神戸三宮店／078(332)3961

● **自由な時間にプログラミング，いかがですか？**

パソコンを買ったのに，プログラムを習うチャンスがなく，市販ソフトだけでがまんしている方，自習書などで勉強しているけれども，よくわからない方，歓迎いたします。

● **実技中心の講座です**

プログラミング技術は，何本も何本もプログラムを作ることにより上達する技術です。当講座は，出来るだけプログラミングをして頂くよう，ワークカセットを用意し，当社に送られてきたプログラムは，プリンタからリストを出力し，添削して返送する出題形式等，工夫をこらしたカリキュラムになっています。

● **幅広い学習ができます**

当社はゲームソフトから簡易言語，データベース，コンピュータ・グラフィックス・ソフトまで開発してきたシステムハウスですから，幅広いプログラミング技術が学べます。

● **プログラム作りには良い道具が必要です**

ご入会の方には，プロが使っている，使い易く，保管し易いＭＤＢＡ特製仕様書類及びバインダー，テンプレートを使って頂きます。良いプログラムを作るには，良い道具が必要です。あなたの部屋も明日からはシステム開発室です。

対象機種　①PC-8001mkII,PC-8001
②FM-7
③X-1
④PC-6001

※機種別の講座ですから，あなたのパソコンにあわせた学習が行なえます。

講座　①ビギナー講座
初めての方、基礎から始めたい方を対象。
②初級講座
FORループが使える方を対象

バグカード　システム
自作プログラムでどうしてもわからないバグにお答えいたします。

※実技中心の講座のため，ご入会はパソコンをお持ちの方に限らせて頂きます。

資料希望の方はハガキに〒，住所，氏名，TEL，機種名を記入の上，当社宛にお送り下さい。早速資料をお送りいたします。

マイクロ データ ベース アソシエイツ
〒194-01 東京都町田市鶴川2-1-8 -205
☎(0427)35-1600

サブルーチンとジェネレータを使った【FM-7、FM-NEW7、FM-77】

F・BASIC ゲーム作りのノーハウ

竹本篤郎監修
ナラシノマイコンクラブ著
定価2300円〒250

UFOやレーザー砲などゲームに登場させる形のパターンジェネレータとゲームに不可欠なサウンド効果を出すサウンドジェネレータのプログラムと数10のサブルーチンを駆使して、ゲームを作るときのテクニックやノーハウを実例で紹介　B5判・208頁

わかる・楽しい・役に立つ

FM-7プログラミング入門

【FM-NEW7、FM-77でも使えます】
※フロッピーディスク頒布サービスあり

高橋三雄著
定価2300円〒250

どこにでもあるような身近な問題を題材にして、プログラムするときの考え方・作り方・発展の方法等をやさしく説明　新しいプログラムへの挑戦・グラフィックスを楽しもう・並べ替えと関数・ミュージックの手ほどき・プログラムライブラリイほか　B5判・240頁

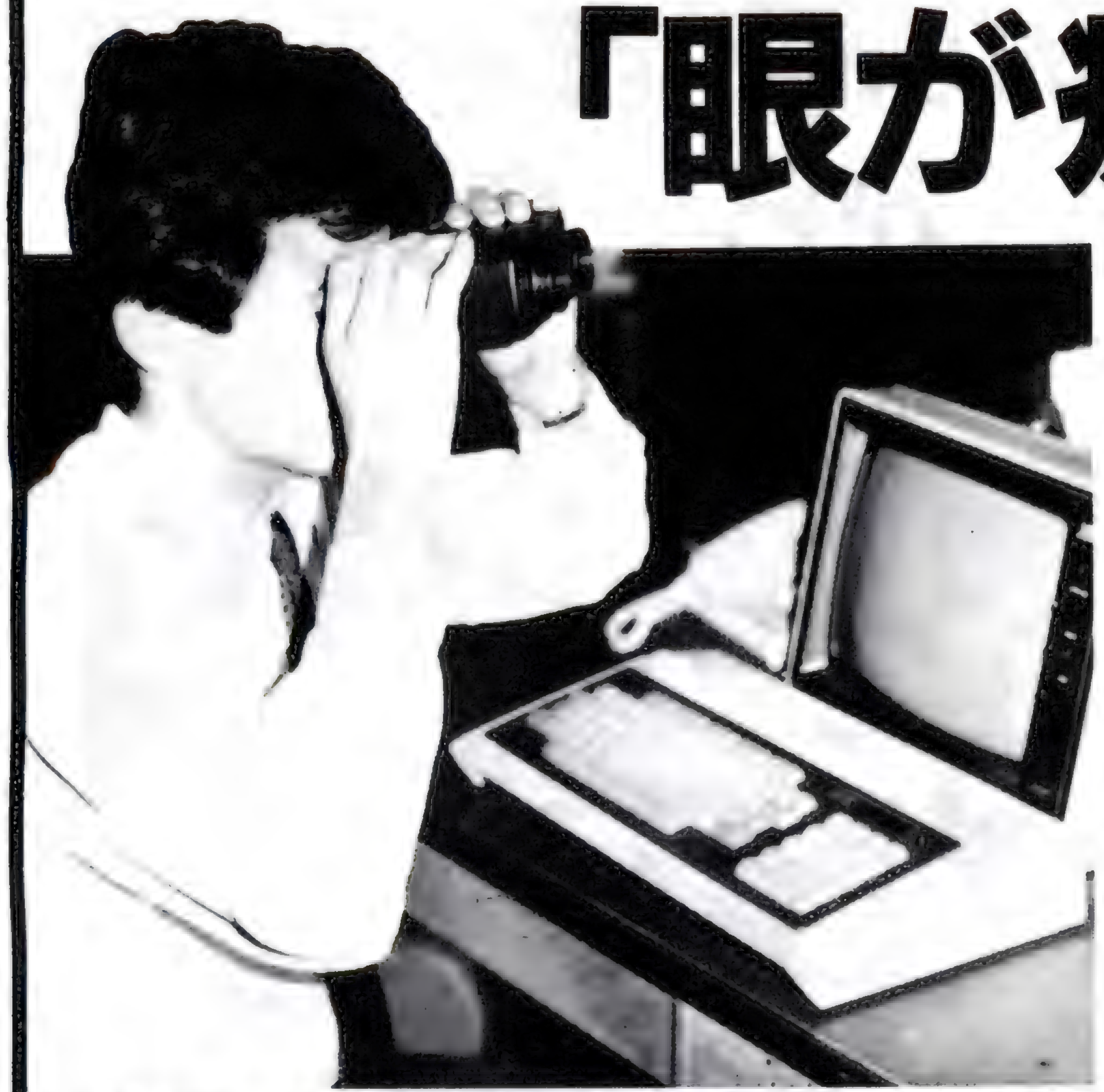
対話形式による【FM-NEW7, FM-77使用可】
FM-7わくわく活用入門
青木　徹著
B5判・226頁
定価2100円〒250
グラフィックを楽しもう・表を作ってみよう・漢字の使い方・活用テクニックルーチン・VIPの活用法・便利な付録ほか

FM-NEW7, FM-77　活用ソフト【FM-7使用可】
ワープロとグラフ
潮　守著
B5判・200頁
予価2300円
近刊
簡易日本語ワープロ・ビジネスクラブ・統計表・統計グラフ・ディスクユーティリティ・オルガン お絵かき・金利計算ほか

山海堂
〒113 東京都文京区本郷5-5-18
振替／東京4-194982　☎03(816)1617

「眼が疲れるなァ…」

大全国反響で



パソコンやワープロの画面を長時間視つづけることは、眼に大きな負担になります。

近視、眼精疲労でお悩みの方に朗報！ 視力低下は病気ではなく、訓練次第で回復可能ということは意外と知られていません。

視力回復には、遠方の凝視、視点移動、中心固視などの訓練が効果的とは科学的にも実証されていますが、毎日、それも長期間続けなければ効果が薄いというのが難点です。

そこで、視力回復と近視予防のために開発されたのが"アイ・トレーニングマシン"。東大名誉教授故石原忍先生の著「小眼科学」に基づいており、遠率7倍の光学レンズとシャッターにより、適度に毛様体筋を運動させる無限焦点式訓練器です。

トレーニングは一日6分。

左の写真のようにながめるだけ。軽量・小型・操作もスイッチボタンひとつという使いやすさで、いつでもどこでもトレーニングできるのが特長です。

近視だからとあきらめてはいけません。

メガネ・コンタクトをご使用の方、ぜひ一度お試し下さい。

医療器具ではありません。目の訓練器です。副作用無し。

★今回、本紙読者に特別頒布！
○定価59,800円を特別価格38,000円で。クレジット月々3,400円より。

只今販売代理店募集中！

○商品お急ぎの方は現金書留にてお申し込み下さい。

お申し込み、資料請求は電話かハガキで下記迄

〒590 大阪府堺市中之町西1―1―29

(株)アイワ OF係

☎0722-22-2047

パワーアップ！ OS-9™

Quality Brains OS-9 Software Series

予期せぬ衝撃！ あのOS-9に珠玉のTOOL新登場！
これで8ビットOSの究極OS-9をフルに使いこなそう。
以下続々新製品開発中、乞御期待。

FMシリーズ用 新発売！

OS-9 Utilities-1 定価12,000円

- ●XferFB（F-BASICのミニ・フロッピィディスクのファイルなんでも転送）
- ●Slist（大きいTEXTFileをスクロールさせながら見れます）
- ●DirALL（あるディレクトリ中のすべてのFileの名を出力する）
- ●Fedit（どんなファイルでもエディトできます）
- ●Hardc（F-BASICよりも速いハードコピー）
- ●Dirlist（デイレクトリのFile名のみを出力）

OS-9 Filters-1 (in BASIC 09) 定価12,000円

- ●Sort（小さなTEXT Fileをメモリー上でソートする）
- ●Pickup（パターンとマッチする行を抜き出す（Grep））
- ●Unique（ソートしたFileの重複行を除く）
- ●Insert（TEXT Fileとコマンドラインを混合してプロシージャファイルをつくる）
- ●TakevarB（BASIC 09のプロシージャからstatement変数を抜き出す）
- ●Blist（BASIC 09のソースをシングルステートメントにして段付けして出力）
- ●WC（TEXTの文字数、語数、行数をカウントする）
- ●Wordcut（TEXTを単語単位に分解する）
- ●Title（TEXTに表題、日付、ページ番号をつけて出力する）
- ●Bprint（BASIC 09のFileをプロシージャごとにページ送りして出力する）
- ●Translit（大文字から小文字へのように一文字単位の置換えをする）

わたし、隠さないわ！
全製品ソースリスト付

製品に関するお問い合せは下記へ。
なお御注文の際には2D/2HDを明記の上、
現金書留か口座振込でお願いします。
銀行口座：第一勧銀・本郷支店
普通075-1500653 クオリティ・ブレインズ

quality brains

文京区本郷2-40-8萩原ビル2F
☎03(818)0024

資料請求券 OH!FM-9

※OS-9とBASIC 09はマイクロウェア社及びモトローラ社の登録商標です

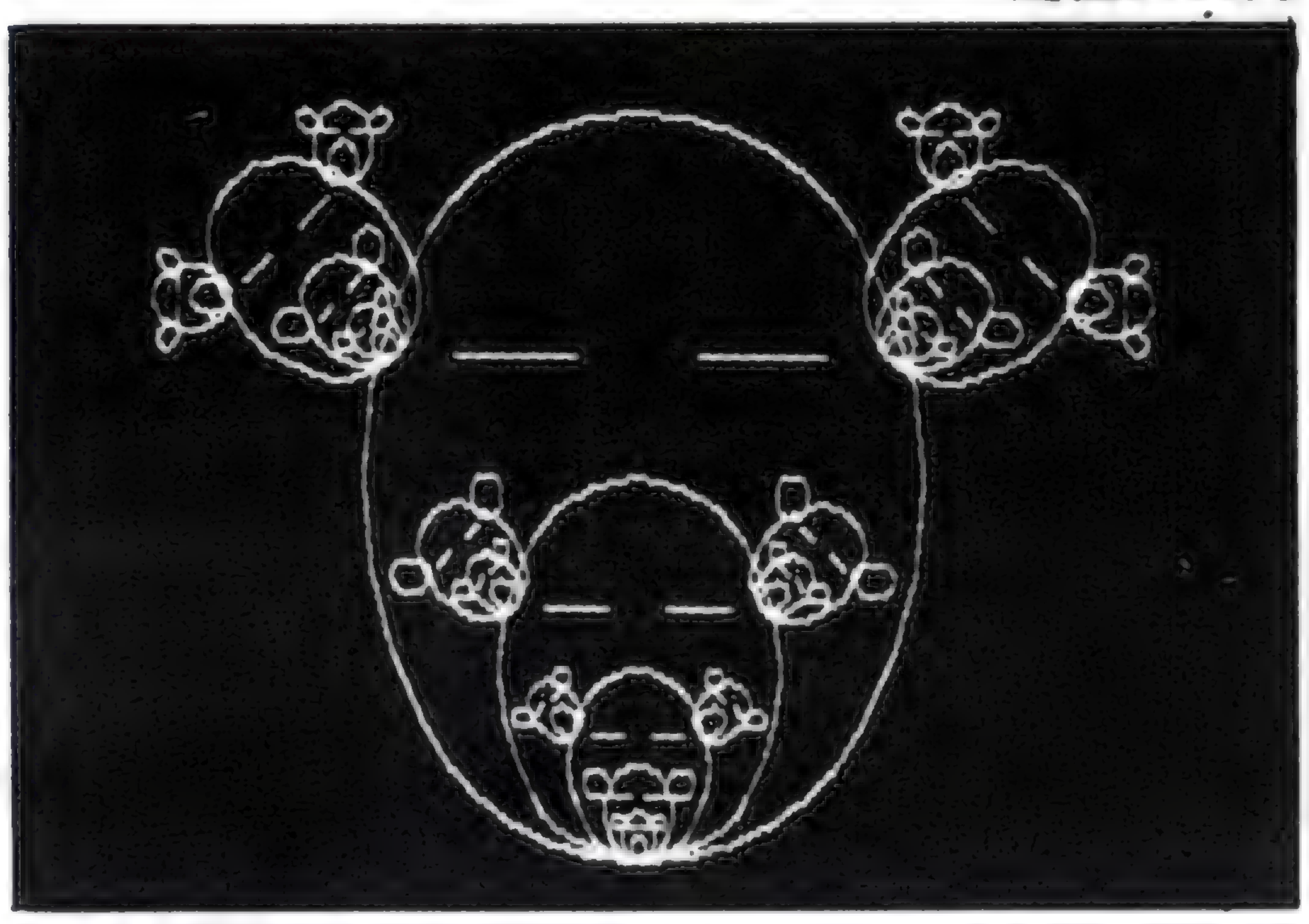
パソコン LOGO プログラミング

FM-7/77用実例プログラムを満載！

小谷善行著　マイコン活用シリーズ　［新刊］

人工知能型言語といわれるLOGOが創造する新しい世界〝Microworld〟を，実例プログラム80本を示しながら，わかりやすく具体的に紹介する。面白い例題，便利なプログラム，BASICによるタートルグラフィックス，勉強になる問題などなど。

定価2400円



パソコン BASIC グラフィックス

新井進・橋本創造著 マイコン活用シリーズ 好評発売中

FM7/8によるグラフィックスプログラム59本を示した入門書。

定価2000円

F-BASIC FM-7／8 グラフィックパッケージ

円グラフ，ヒストグラム，3次元折れ線グラフ，日本分県地図，3次元イメージプロセッサー，自由線ドットライター，ロビーデザイン・シュミレーターの7本の実用プログラム。

定価5000円(〒500円)

大好評の赤い本！ 続々刊行

パソコンBASICプログラミング
松尾守之・宍吹雅敏 著　1800円

パソコンビジネスソフト活用法
新井 進・金田浩邦 著　2000円

パソコンBASIC数値計算Ⅰ
小島紀男・町田東一 著　2800円

パソコンBASIC数値計算Ⅱ
町田東一・小島紀男 著　2800円

パソコンBASIC統計解析
大滝・鈴木・長澤著　2800円

東海大学出版会　〒160 東京都新宿区新宿3-27-4 東海ビル 電(03)356-1541

FMの専門教育機関で 明日からテクノ人に

16Bitパソコンによる授業風景

パソコンコース

各コース共・毎月開講
FM-11 24台

コース	対象	内容	授業料
入門	パソコン未経験者	パソコンの基本概念 BASICの基本命令	20,000円
演習	入門コース修了か基本操作をマスターしている方	基本命令の徹底演習 シーケンシャル、ランダムファイルのプログラム作成	20,000円
実務応用	演習コース修了かパソコンを実務レベルで使いたい方	市販ソフトの操作・修正 （財務会計、給与計算、簡易言語、漢字ワープロ）	20,000円
短期集中	短期間でパソコンの操作をマスターしたい方	月～土の一ヶ月間 いつでも受講できます。	50,000円

ワープロコース

フレックス制採用
OASYS 100J 44台

コース	対象	内容	授業料
速習	ワープロ未経験者	ワープロの基本概念から「練習システム」を用いたキー操作	20,000円
オーディオワープロ	速習コース修了者または同等のレベルの方	オーディオ（口述筆記）による速度練習や応用手法	40,000円

毎月1日　無料説明会

詳細は案内書(無料)を請求して下さい。

名門 短期合格のTOKYO ACCOUNTING COLLEGE

学校法人 東京会計専門学校

〒130 東京都墨田区錦糸2-13-7　TEL 03(624)5441(代)

キーボードカバー

実用新案登録出願中
意匠登録出願中

SOFTOP®

イニシャルシートプレゼント

材質：高級アクリル

写真はFM-77用

パソコン用（各）3,800円

- ❸ ★ FM-New7
- ❸ ★ FM-8
- ❶ ▲ FM-11
- ❶ ▲ FM-11BS
- ❶ ▲ FM-11AD2
- ❶ ★ FM-77

OA用　4,800円

- ❶ ▲ 9450

MSX用 3,800円

- ❶☆★ FM-X

他に、NEC、シャープ、日立
パナファコム、ナショナル、三菱
ソニー、ソード、沖電気、東芝
セガ、IBM、アップル
など各社キーボードカバーも製
造しています。

カラー	
☆	透　明
★	ブラウン・スモーク
▲	グレー・スモーク

※多量の特別注文は御相談ください。

❶タイプ：キーボードのまわりまでかぶさるタイプ
❷タイプ：キーボードの一部がかぶさるタイプ
❸タイプ：キーボードの上にのせるタイプ

マイコン用　（型番 TC-10）

宛名 タックシール

- 4インチ幅トラクタフィールド用紙
- 1ケース・1ロール500枚入×2コ入
- シールサイズ(タテ)38㎜×(ヨコ)86㎜

¥3,200

フロッピーディスケット

通信販売専用（この商品のみ送料500円かかります）

マクセル/MD2D	10枚で ¥8,500	5枚で ¥4,500
フジ/MD2D	10枚で ¥8,500	5枚で ¥4,500

■お求めは……(フロッピーディスケット以外の商品送料はサービスいたします)
全国パソコンショップまたは直接右記へお申し込みください。
- 現金書留でご注文の場合は商品名、住所、氏名、電話番号をはっきり書いてお送りください。
- 銀行振込みでご注文の場合は商品名、住所、氏名、電話番号をハガキまたは電話でご連絡ください。

発売元
エイト電気株式会社　FM係
〒110 東京都台東区上野5-3-4 ☎03-831-5632(代)
振込銀行：北陸銀行 上野支店　(普)No.4032110
三栄電子株式会社　FM係
〒556 大阪市浪速区日本橋東2-10-2 ☎06-643-3833(代)
振込銀行：三和銀行 恵美須支店　(普)No.189879

新発売

FM-7の性能を極限まで利用
の本格的トータル学習システム。

FM-若葉塾

〈使用言語〉BASIC＋マシン語

こんな学習ソフトがほしかった！

これまでのカセットテープ版にありがちだった単純なデータの繰り返しをランダムアクセスファイルを用いることにより解消し、学習内容を変化の富んだものにしています。

「英単語学習」の中1～中3は、日本語訳・英語訳・アクセントの3通り。受験用は単に日本語訳・英語訳にとどまらず、反対語・名詞・形容詞・副詞・動詞の語形変化を多角的に学習できるようになっています。

「英文法学習」は、「重要事項のまとめ」、「練習問題」、「テスト」の3段階に分けて学習。また文法事項も選択できるので、学校の進度や学力に合わせて学習できます。

「重要文型学習」は、重要な熟語等を含む文章について、整序問題や空所補充問題で学習し、学習するごとにそれまで間違えた問題をくりかえし、復習します。

お申込方法：商品名、住所、氏名、電話番号を明記して、下記の株式会社一夢堂宛に現金書留でお申込ください。

■制作　若葉塾　　■発売元 株式会社 一夢堂 ソフト開発課
〒110 東京都台東区東上野1-26-8 TEL.03-832-0151

FM-7 5インチディスク版 要：漢字ROM
FM-77 3.5インチディスク版

各学年 ¥6,800
受験用 ¥7,800

（各フロッピーディスク1枚、説明書付）
送料は1本につき¥200です。

FM-若葉塾シリーズ

- 中学1年 英単語学習
- 中学2年 英単語学習
- 中学3年 英単語学習
- 受験用 英単語学習
- 中学1年 英文法学習
- 中学2年 英文法学習
- 中学3年 英文法学習
- 受験用 英文法学習
- 中学1年 英語重要文型
- 中学2年 英語重要文型
- 中学3年 英語重要文型
- 受験用 英語重要文型
- 中学1年 数学 (開発中)
- 中学2年 数学 〃
- 中学3年 数学 〃

※英単語学習の受験用には「でる単作成ソフト」が付いています。

資料請求券 oh!FM9

EDITORS' ROOM

◘オリジナルプログラムを下記の要領で募集いたします。
- プログラムはテープまたはディスケットに入れてください。
- 原稿およびテープなどにも氏名・住所を明記してください。
- プログラムは未発表のものに限ります。また他誌との二重投稿，盗作などモラルに反する応募は固くお断りします。
- 応募原稿は原則としてお返し致しません。ディスケットは，新しいものをお返しいたします。
- 原則として受取通知のみいたします。採用／不採用通知はいたしません。
- 掲載にあたって多少修正させていただくことがあります。
- 掲載の場合は本誌規定の原稿料をお支払いいたします。
- 応募者全員にOh／FMシャープペンシルを進呈いたします。

〒102 千代田区四番町2-1 ㈱日本ソフトバンク出版部 Oh／FM編集室

◘**添削用プログラム募集**

プログラムを組んでみたが，自信がない，もう少しスマートに組むにはどうすればよいかわからない。こんな方は，ぜひ当編集部にお送りください。誌上にて添削いたします。
- プログラムは，必ずテープまたはディスケットに入れてお送りください。
- 使用機種，プログラムの概要，キー操作，住所，氏名など，必要事項は明記してください。
- 添削は誌上のみです。掲載されないものについて，個別に添削することはございませんので，ご了承ください。
- 添削原稿は，お返し致しません。
- あて先は，Oh／FM編集室添削教室係です。

作品のアイディアは浮かんだが，BASICテクニックは自信がない，という方，どしどしご応募ください。お待ちしております。
- 本誌掲載プログラムの打ち込みミスを添削するものではありません。

◘Oh／FM**協力スタッフを募集**しています。電話のうえ，実力評価用の作品（ソフトまたはハード）をご持参ください。

Oh／FM編集室 松岡 ☎03(261)4095

◘本誌の内容に関して，電話による質問を受け付けます。

☎03(265)5789 受付時間 月～金 16:00～18:00

質問はバグ情報についてのみ受け付けます。機械の取り扱いなどについてのお問い合わせはご遠慮ください。

▶**広告索引**（五十音順）

I・Oデータ機器……159
一夢堂……175
インテック……158
エイト電気……175
エス・ビー・シー……162・163
エプソン……9
エニックス……21
エム・アイ・エー……167
エム・エー・シー……10
キャリーラボ……20
クオリティ・ブレインズ……173
光栄……164・165
コスモ製作所……157
コマス……8
コミネ・エージェンシー……161
コンピュータ・イレブン……171
山海堂……172
システムセンター三重……170
シャノアール……30
秀和システムトレーディング…14・15
シルバーボール……28・29
鈴木技研……155
住友スリーエム……13
精工舎……表3
ソニー……16・17
ソフトプロインターナショナル…23
ソフトマート……156
ダイナミック……154
九十九電機……160
T&Eソフト……19
東海大学出版会……174
東京会計専門学校……174
日本近視回復センター……173
日本ファルコム……18
服部セイコー……表3
ハドソンソフト……24～27・168・169
富士通……表2・表4・1～3
富士通マイコンシステムズ……4・5
ブラザー……11
ボーステック……22
ポリシー……12
マイクロデータベースアソシエイツ……172
マクロシステムズ……166
理工産業……154

▶編集後記

■最近編集部内で，まちがいだらけの表紙さがしというのがはやってまして困っとるんですよ。新しい表紙を作ってもってくると，あれ，この影変じゃない？とか，このストロー，水で屈折してみえるはずじゃないの？ とかうるさいんですよ。あげくのはてに，なんでお盆の上にマウスがのってるの？とか，木の枝がくさりをしてるとか言いだすんですよ。みなさん，たすけてください。㋐

■アメリカ・ミネアポリスで開かれていたSIGGRAPH'84も終わり，次回はサンフランシスコで開かれる。今年のSIGGRAPH期間中は軽井沢でテニスをしていたというていたらく。来年こそは絶対見に行く。だからMさん，仕事ください。 ☺

■人生を生きるということは，巨大なプログラムを組み上げるようなものである。今日もエラーに悩まされ，デバックに取り組むのであった。合掌…（無人）

■何となくバンクに来て，早や半年がすぎてしまいました。大学の方が忙しいので……。ゴメンナサイ……。 （蓉）

■海の近くに住んでいるH君は，夏になるとバンクで避暑をするのが恒例だそうです。あんな汚い海，誰だ，湘南をメチャクチャにする奴は！ オニイサンは，おこっているのダ…。 (蓉)

■暑い猛暑の中，四番町の某出版社に着くと，冷房からの涼しい涼風が直撃するマシン室に入っていく。その後はいつ日が日没したのかも気付くことなくその日最後の終電の時刻がせまる頃まで仕事……なんてことは表向きで，いつもみんなで仕事をうっちゃらかしてどこかに遊びにいくのが僕ら協力スタッフの常です。なお先月，名なしになったことで印刷の人を恨んだりしません。 ろ

■情報化・電子化の波が世界的に進んでいるとよく言いますが，わりと外国ネタにはうといものです。たまには，大きな書店などの外国書売り場をのぞいてみませんか。byte や韓国語のブルーバックスもながめて面白いものですが，きわめつけは中国語の工学書。電脳だの申子，半導體などの本には，おなじみの用語が漢字で出ています。ちょいとメモして雑談のネタにどうですか。（狂）

■髪を伸ばしはじめて早や４ヶ月。ようやく歌舞伎町などを歩いていても「お兄さん，安いよ」と言われなくなりました。呼び込まれなくなったのはいいけれど，以前は誰でもくれたそのテの店の宣伝ティッシュペーパーがもらえなくなってしまった。おかげで今は，ティッシュペーパー欠乏症です。 （妙）

■数年ぶりに大学へ。研究室も様変わり。よくよくダベッた同級生，今じゃ子持ちの大親分。そしてパソコン稼動中。10年前のミニコンに，優るも劣らぬ高性能。データの解析，計測にアッと驚く大活躍。まだまだオジンじゃないけれど，時のたつのは速いもの。かつてパソコンあったなら，今ごろ小生もオーソリティ（？） （一）

■１＋１＝０，２＋４＝０，６＋13＝１，19＋20＝２，100＋100＝４，では18＋20はいくつ？ この問題は幼稚園児や小学生ならすぐに解けるそうです。ORやAND，ex-OR？ それとも12進数？ などと考えてなかなか解けない人は，もっと単純にものごとを見つめる目を養いましょう。つまり，私のことですが……。 （M）

・編集 松岡真理 真部一善 渡辺妙子 八重樫和也
・技術 松田辰夫 小林初雄 徳永 聡
・海外協力 SHIGEYOSHI KOMAGATA
・協力スタッフ 川西慶一 菊池蓉子 木下淳博 志沢雅彦 末安直樹 鶴岡哲明 長沼孝仁 西村義孝 平野 剛 古田 昌 本郷誠司 米村英明 篠原孝夫
・写真 浜崎 昭 杉山和美
・アートディレクタ 藤林省三
・レイアウト オメガ社
・表紙CG Tea Spoon（木下淳博）
・イラスト 高橋正輝

月刊 Oh!FM 9月号

・昭和59年９月１日発行
・発行人／孫 正義
・編集人／田鎖洋治郎
・発行元／株式会社日本ソフトバンク
本社／〒102 東京都千代田区九段南２-３-14 靖国九段南ビル2F
☎03-263-3690 Fax. 03-263-3660
出版部／〒102 東京都千代田区四番町２-１ ☎03-261-4095
大阪支店／〒542 大阪市南区難波千日前５-19河原センタービル
☎06-644-0191 Fax. 06-644-0160
・印刷／図書印刷 ☎03-453-2550
・海外送付ご希望の方は下記へお問い合わせください。
日本IPS株式会社／〒101 東京都千代田区神田小川町３-５ ☎03-291-2632(直)

©1984 JAPAN SOFT BANK 雑誌02199-9
★本誌からの無断転載を禁じます。

SHOSEN

読者の皆さまのご意見をお寄せください。誌面づくりに反映させるための貴重な資料となりますので、ご協力ください。

料金受取人払

郵便はがき

102

麹町局承認
161

差出有効期間
昭和60年3月
14日まで

(受取人)

東京都千代田区
四番町2-1

㈱日本ソフトバンク

Oh!FM 編集室行

プレゼントNo.

整理番号 9 *

氏名	年齢	性別	マイコン歴
		男　女	年　　か月

住所 〒　　　都道府県

☎

職業・勤務先
学校・学部・学年

本誌の購読は	はじめて，第　月号から，ときどき，毎号	本誌以外によく読む雑誌

＊印の欄には記入しないでください。

Oh!FM 9月号

愛読者アンケート ⑨

○特集について

非常によかった　よかった　普通　よくなかった　ざんげせよ

(理由　　　　　　　　　　　　　　　　　　　　　　　　)

○ 感想，要望，READERS' AREAへ一言，そのほかなんでもお書きください

○いちばん良かった記事

○載せてほしい記事・内容を具体的に

○どんなパソコンをお持ちですか　(個人用　会社等　ない)

FM-7，NEW7，77，8，11(EX, AD, ST, BS, AD2)，X，他(　　　　　)

CRT〈グリーン・カラー〉(メーカー名　　　　　　　　　　　)／家庭用

Dレコーダ(メーカー名　　　　　　　　　　　)／ラジカセ

ディスク〈S・D〉(型番 メーカー名　　　　　　　　　　　)

プリンタ(型番 メーカー名　　　　　　　　　　　)

漢字ROM　Z-80カード　RS-232C　マウス　音声合成

その他(　　　　　　　　　　　　　　　　　　　　　　　)

○ソフトを自作したことがありますか

(本数＝　　言語＝　　ステップ数＝
ジャンル＝　　用途＝　　)ない

○今，一番気にいっている市販ソフトは

ゲーム(ソフト名 発売元　　　　　　　　　　　)ない

その他(ソフト名 発売元　　　　　　　　　　　)ない

読者の皆さまのご意見をお寄せください。誌面づくりに反映させるための貴重な資料となりますので，ご協力ください。

切り取り線

各票の※印欄は、払込人において記載してください。

通常払込料金加入者負担

払込通知票

口座番号	東京 1	2	9	3	0	7	金額	¥6000
加入者名	株式会社 日本ソフトバンク						料金	払込み / 特殊 円
払込人住所氏名	※ (郵便番号)						備考	
							受付局日付印	

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。 (郵 政 省)

記載事項を訂正した場合は、その箇所に訂正印を押してください。

切り取らないで郵便局にお出しください。

通常払込料金加入者負担

払込票

口座番号	東京 1	2	9	3	0	7
加入者名	株式会社 日本ソフトバンク					
金額	¥6000					
払込人住所氏名	※					
備考	受付局日付印					

振替用紙

←点線から、きれいに切り取ってご使用ねがいます。

切り取り線

送り先	お名前	フリガナ	性別	年齢	ご職業
			男・女		
	ご住所	フリガナ 〒			
	お電話	ご自宅	勤務先		
申込書	Oh!FM 定期購読	新規申し込み 84年　　月号より１年間	年間（12冊） ￥6,000円(送料共)		
	注意事項　このお申し込みに別冊・増刊は含まれませんのでご了承ください。なお，継続延長のお申し込みは契約期限の切れる１か月前までにお願いします。				
通信欄					

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。　（郵　政　省）

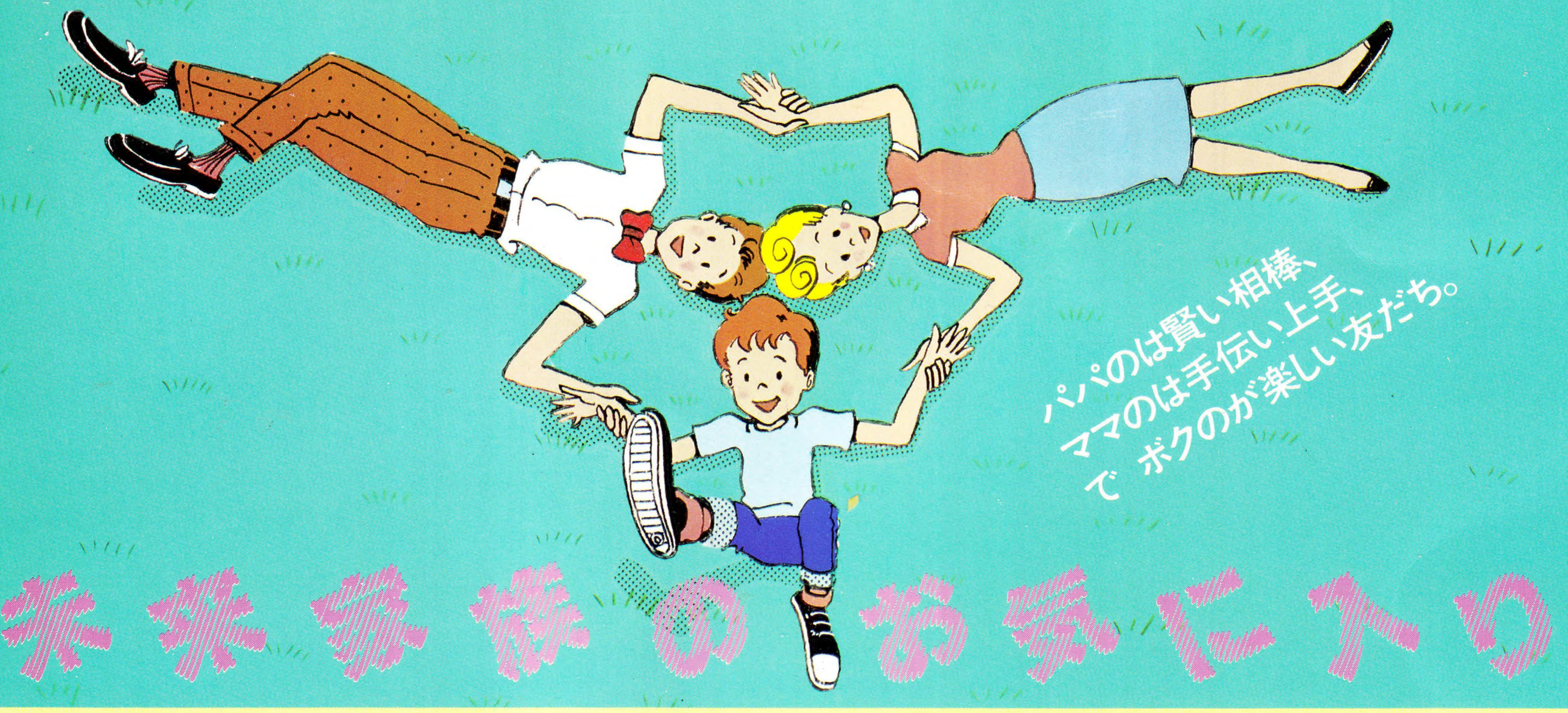
この欄は、加入者あての通信にお使いください。

切り取らないで郵便局にお出しください。

本誌をご愛読いただきありがとうございます。バックナンバーのご注文はお近くの書店からできますが、どうしても手に入りにくい場合、直接弊社へ現金封筒にて注文してください。

切り取り線

SEIKOSHA



●文字種類は、英数、カナ、記号160種、画素64種●タテ8ドットによるグラフィック印字、グラフィックデータの繰り返し印字が可能。●1/6インチ、またはプログラム指定によりn/72インチ改行が可能。●倍密度グラフィックの印字が可能。●適応機種FM-7、8に対応。

FMシリーズ対応
マルチ機能
搭載プリンタ

¥49,800

Graphic Printer
GP-500F

GP-550E

¥119,800

多目的
漢字プリンタ

●PCシリーズに対応、FMシリーズにはROM交換で対応可能
●FMシリーズ用ROM
＃**GP-55880**　¥5,000
●NEC純正漢字ソフト用ROM
＃**GP-55881**　¥5,000

GP-700M

¥99,800

高品位
カラープリンタ

●PCシリーズに対応、FMシリーズにはROM交換で対応可能
●FMシリーズ用ROM
＃**GP-70880**　¥5,000
●オプションビデオインタフェースにより、PC-8001、PC-8001MKⅡ、PC-8801のカラーハードコピーが可能
＃**GP-70060**　¥16,000

GP-50MX

新発売

MSX対応
コンパクトプリンタ

GP-50MXセット販売セール実施中

期間/昭和59年8月1日～9月30日
好評のMSX対応コンパクトプリンタGP-50MXに、ケーブルと画面コピーソフトをセットしました。

セット価格 **¥35,000**

単品価格：本体 ¥29,800　ケーブル ¥6,000　画面コピーソフト ¥3,000
※MSXマークは、マイクロソフト社の商標です。

●GP-700M、GP-550Eの「プリンタ使いこなし読本」発売中 ¥1,700
●GP-700M アップル用インタフェース(カラーハードコピー可能)＃GP-70015発売中 ¥29,800
※詳しい問合わせ先：セイコーシャ情報機器販売㈱ 大阪/06(229)0206 東京/03(861)2786

株式会社 服部セイコー 特品事業部 情報機器部
〒101 東京都千代田区鍛冶町2−1−10 服部セイコー神田別館 TEL 03(256)2111

株式会社 精工舎 情報機器事業部 営業部営業課
〒130 東京都墨田区太平4−1−1 TEL 03-623-8111(代) 03-625-4610(直)

資料請求券
Oh! FM 9

Oh!FM 1984 9月号

昭和五十九年九月一日発行 第二巻第七号通巻十三号
昭和五十九年六月八日第三種郵便物認可 昭和五十九年二月二十八日国鉄首都特別扱承認雑誌第七四〇三号

Oh!FM 1984 9

●特集=どれにしようか OSいっぱい
●簡易ジョイスティックの製作
●グラフィックジェネレータ

日本ソフトバンク

Oh!FM ㈱日本ソフトバンク発行 Printed in Japan 定価480円 雑誌 02199-9